AFRL-SN-WP-TR-2004-1255

MAXIMUM A POSTERIORI (MAP) ESTIMATES FOR HYPERSPECTRAL IMAGE ENHANCEMENT

Gregory L. Wilson, Andrew C. Lindgren, Thomas M. Fitzgerald, and Pamela S. Smith

Alliant TechSystems – Mission Research Corp.
Dayton, OH 45430-2108

Russell C. Hardie

University of Dayton
Department of Electrical and Computer Engineering and Electro-Optics Program
Dayton, OH 45459-0226

SEPTEMBER 2004

Final Report for 30 March 2002 – 30 May 2004

THIS IS A SMALL BUSINESS INNOVATION RESEARCH (SBIR) PHASE II REPORT.

STINFO FINAL REPORT

SENSORS DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320

NOTICE

USING GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA INCLUDED IN THIS DOCUMENT FOR ANY PURPOSE OTHER THAN GOVERNMENT PROCUREMENT DOES NOT IN ANY WAY OBLIGATE THE U.S. GOVERNMENT. THE FACT THAT THE GOVERNMENT FORMULATED OR SUPPLIED THE DRAWINGS, SPECIFICATIONS, OR OTHER DATA DOES NOT LICENSE THE HOLDER OR ANY OTHER PERSON OR CORPORATION; OR CONVEY ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY RELATE TO THEM.

THIS REPORT HAS BEEN REVIEWED BY THE AIR FORCE RESEARCH LABORATORY WRIGHT SITE OFFICE OF PUBLIC AFFAIRS (AFRL/WS/PA) AND IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONALS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

/s/

Philip S. Maciejewski, Program Manager
E-O Threat and Targeting Detection Tech. Branch
Sensors Directorate

/s/

Barry K. Karch, Acting Chief
E-O Threat and Targeting Detection Tech. Branch
Sensors Directorate

/s/

ROBERT D. GAUDETTE, Colonel, USAF
Chief, E-O Sensor Technology Division
Materials Integrity Branch
Sensors Directorate

Do not return copies of this report unless contractual obligations or notice on a specific document require its return.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
September 2004	Final	03/30/2002 – 05/30/2004

4. TITLE AND SUBTITLE
MAXIMUM A POSTERIORI (MAP) ESTIMATES FOR HYPERSPECTRAL IMAGE ENHANCEMENT

5a. CONTRACT NUMBER
F33615-02-C-1169

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
65502F

6. AUTHOR(S)
Gregory L. Wilson, Andrew C. Lindgren, Thomas M. Fitzgerald, and Pamela S. Smith (Alliant TechSystems)
Russell C. Hardie (University of Dayton)

5d. PROJECT NUMBER
3005

5e. TASK NUMBER
11

5f. WORK UNIT NUMBER
TM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Alliant TechSystems – Mission Research Corp.
Dayton, OH 45430-2108

University of Dayton
Department of Electrical and Computer Engineering and Electro-Optics Program
Dayton, OH 45459-0226

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Sensors Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson AFB, OH 45433-7320

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/SNJT

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-SN-WP-TR-2004-1255

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Report contains color. This is a Small Business Innovation Research (SBIR) Phase II report.

14. ABSTRACT

In completing this Phase 2 SBIR, ATK Mission Research with help from the University of Dayton, has accomplished the intended goal of the SBIR program by developing a commercially marketable Hyperspectral Image Enhancement system. This system implements a coherent and rigorous physics model for HSI spatial resolution enhancement on a Mercury Multi-Computer platform.

Our SBIR research effort has led to the completion of a Maximum A Posteriori (MAP) algorithm for constructing a high-spatial resolution hyperspectral enhancement from a low-spatial resolution hyperspectral image, by incorporating registered higher-resolution imagery from an auxiliary sensor. Although we have focused on the enhancement of a hyperspectral image using high-resolution panchromatic data, the estimation software delivered allows for any number of spectral bands to be employed from the auxiliary sensor.

15. SUBJECT TERMS
Hyperspectral, Resolution, Broadband, Maximum A Posteriori, Covariance, Stochastic Mixing, Principal Components, Parallel Processing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Philip S. Maciejewski
Unclassified	Unclassified	Unclassified	SAR	144	19b. TELEPHONE NUMBER *(Include Area Code)* (937) 255-9902 x4360

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Contents

List of Figures

List of Tables

Chapter 1

Executive Summary

Hyperspectral Imagery (HSI) remains the most advanced tool for passive remote sensing. A hyperspectral data cube can contain a vast range of spatial, spectral, and temporal information. Interest in Hyperspectral imagery is creating markets for such applications as military intelligence, tactical military operations, geology, forestry, cultural studies, and environmental studies. These disparate remote sensing communities require sophisticated hyperspectral techniques to solve intrinsically difficult problems such as automated local area change detection, small target detection in clutter, material identification at the sub-pixel scale, target class discrimination for targeting, and geophysical or biochemical parameter estimation for natural resource development or environmental monitoring.

However there are limitations. Hyperspectral imaging systems are usually of lower spatial resolution than multi-spectral or panchromatic (broadband) imaging systems. The low to medium spatial resolution of hyperspectral imagery limits its application to real-world problems such as scene spatial change detection or image analysis.

In completing this Phase 2 SBIR, ATK Mission Research with help from the University of Dayton, has accomplished the intended goal of the SBIR program by developing a commercially marketable Hyperspectral Image Enhancement system. This system implements a coherent and rigorous physics model for HSI spatial resolution enhancement on a Mercury Multi-Computer platform. While initial development was accomplished using the 8-node Mercury AdapDev 1280 computer system, we have demonstrated that our code easily scales to larger vector processing clusters such as the 96 (Titan) and 64 (Hawk) node Mercury systems residing at the Wright Patterson Air Force Base Computer Facility.

Our SBIR research effort has led to the completion of a Maximum A Posteriori (MAP) algorithm for constructing a high-spatial resolution hyperspectral enhancement from a low-spatial resolution hyperspectral image, by incorporating registered higher-resolution imagery from an auxiliary sensor. Although we have focused on the enhancement of a hyperspectral image using high-resolution panchromatic data, the estimation software delivered allows for any number of spectral bands to be employed from the auxiliary sensor.

The technique is suitable when some correlation exists between the auxiliary image and the image being enhanced. For example, LWIR HSI may be improved using MWIR broadband imagery as shown in Figures 1.1 and 1.2. Using principle component analysis, we have shown that our MAP estimator produces a hyperspectral image with enhanced sub-pixel content evident not only in the first principal component image, but in lower components as well. Typical principal component substitution methods seek to enhance only the first principal component and do not enhance lower components at all. Any enhancement of these low-signal lower components indicates a promising result.

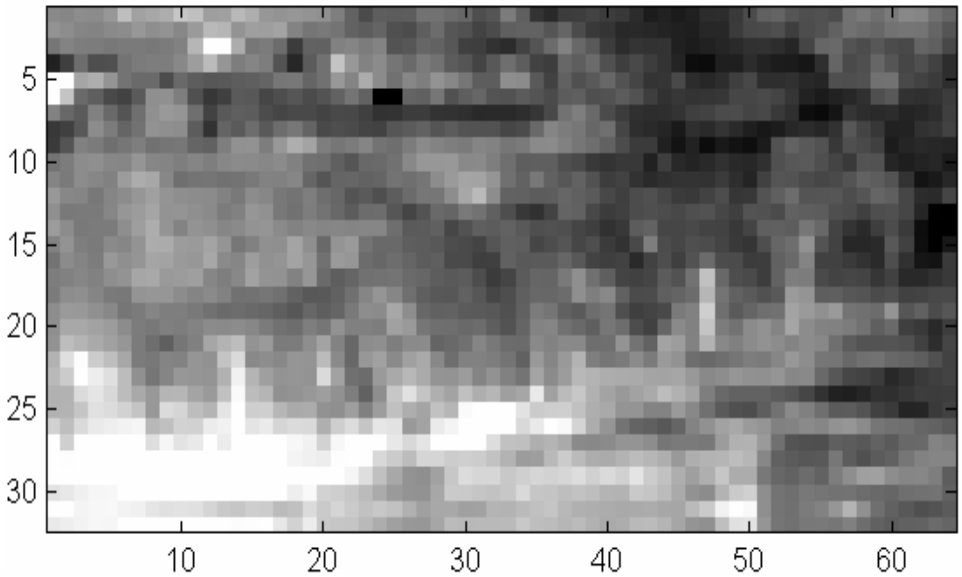


Figure 1.1: SEBASS imagery: band 50 LWIR 4x degraded resolution (10.37 m)

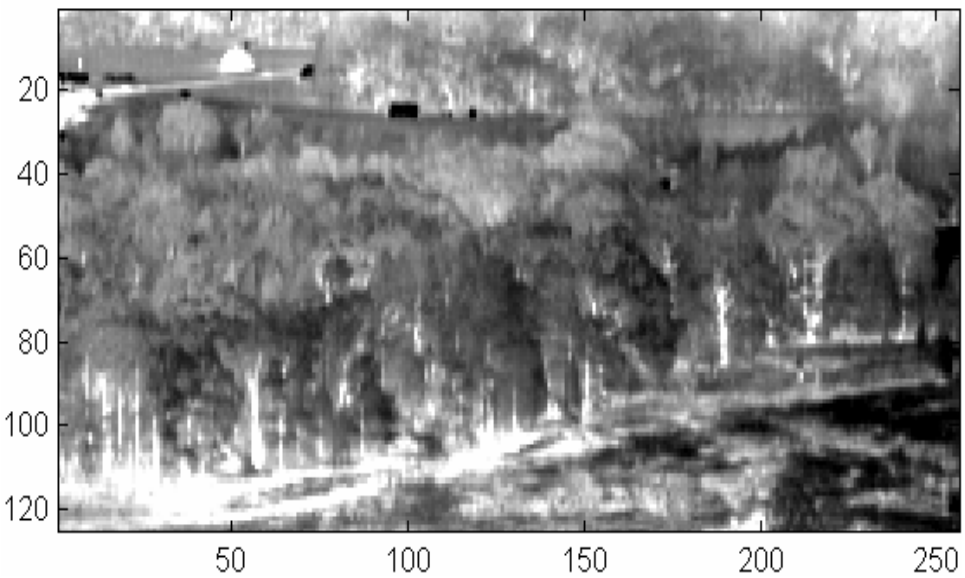


Figure 1.2: SEBASS imagery: MWIR broadband

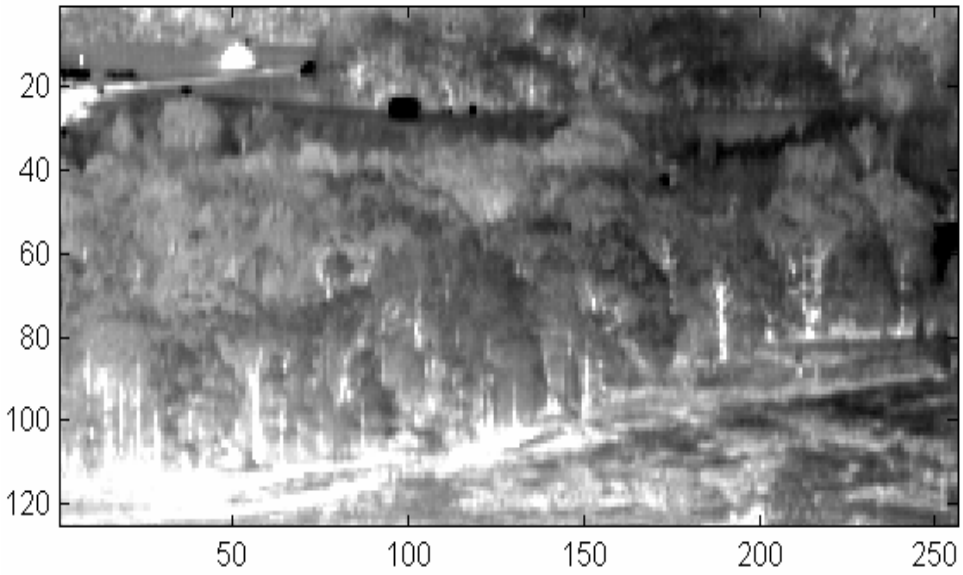


Figure 1.3: SEBASS imagery: MAP estimate of band 50 LWIR using MWIR broadband imagery

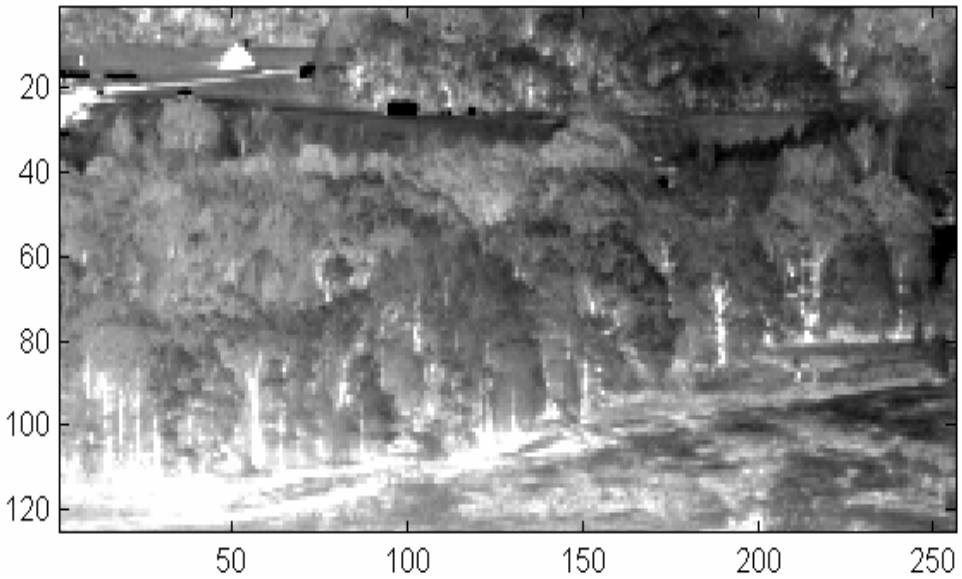


Figure 1.4: SEBASS imagery: True band 50 LWIR

We believe that our MAP approach represents the best results from the most theoretically sound method proposed to date for merging hyperspectral imagery with an auxiliary sensor. In any applications where panchromatic sharpening is used for human interpretation, our method offers improved performance. A technical strength of our MAP estimation framework is that it allows for the incorporation of alternate methods of parameter estimation into the existing code. This modifiable estimation framework will allow AFRL/SN to explore new methods for estimating statistical parameters which hold the promise of still better image enhancement.

The original proposal objectives follow. We believe that we have successfully addressed the tasks defined in the initial proposal and outlined below. As the tasks evolved, we reassigned resources to Objective 1 in order to meet the primary emphasis of the contract, to develop the prototype system.

Objective 1: Develop a prototype near real time hyperspectral resolution enhancement system that implements the algorithm developed during Phase I.

Complete: This task represented 80% of our technical effort. The Mercury 1280 AdapDev system we are delivering is a prototype multi-computer resolution enhancement system incorporating the final and improved version of our algorithm.

Objective 2: Further develop the MAP estimator to take advantage of any a priori spectral information such as known spectral signatures in the scene.

Complete: The final algorithm incorporates capabilities for incorporating user-selectable multi-spectral imagery and for outputting Principal Component imagery. We have also included the capability for users to incorporate their own linear observation models for combinations of endmember spectra.

Objective 3: Incorporate techniques for Temperature/Emissivity Separation (TES) for calibrated high-resolution hyperspectral data. The goal is to recover land surface temperature from the calibrated resolution enhanced hyperspectral data.

Complete: Although not implemented in the final product, a technique was researched and detailed in Appendix C. It is anticipated that future customers will have their own favored TES algorithms that we can implement as required.

Objective 4: Incorporate multi-frame enhancement techniques to be applied to the broadband imagery prior to hyperspectral fusion.

Complete: Again, our final algorithm includes the capability for users to incorporate their own linear observation models for combinations of endmember spectra. We have not specifically included multi-frame enhancement techniques in the final product as most hyperspectral and high-resolution panchromatic imaging systems incorporate scanning technology rather than focal plane arrays suitable for multi-frame enhancement.

Objective 5: Incorporate Air Force Research Laboratories atmospheric radiance and transmission models to allow atmospheric correction for raw hyperspectral data.

Complete: During this task we ran a NASA/JPL parallelized version of MODTRAN 3 on our multi-node Beowulf Cluster. We stopped short of fully incorporating parallel FLAASH code into the AdapDev due to funding constraints. Our implementation plan was to first operate the parallel FLAASH code on a multi-node Beowulf cluster and then to migrate the code to the Mercury platform. We still believe that this is a feasible approach and propose to investigate it as a follow-on step.

Objective 6: Incorporate image registration capability in the prototype to align low-resolution hyperspectral imagery with high-resolution broadband imagery.

Complete: We started this task with a simple registration error analysis. Our results demonstrated that if the panchromatic auxiliary image is perturbed by even 1 full pixel from the HSI data our MAP algorithm is no longer any more effective than simple interpolation. Even a registration error of a fraction of a pixel degrades performance substantially. Because the MAP algorithm is so sensitive to registration we conclude that our Image Enhancement product is best suited to improving HSI collocated with the auxiliary sensor and sharing common viewing optics. This platform architecture will reduce registration error upfront.

Future Work

1. Explore other forms for the critical MAP algorithm conditional covariance matrix and imply other assumptions regarding the nature of the high-resolution hyper-pixels. There is on-going work in the area of Blind Source Separation that may help with the hyperspectral pixel unmixing problem.

2. Develop a covariance matrix partitioning scheme that increases MAP algorithm performance, subject to speed and memory tradeoffs. The results will depend on the specific scene used and the relative quantities of the spectra present.

3. Incorporate the parallelized version of the atmospheric correction code FLAASH into the MAP enhancement algorithm on a Mercury platform.

Chapter 2

Introduction and Overview

The inherent trade-off between spatial and spectral resolution in hyperspectral sensors has prompted the development of remote sensing systems that include low-resolution hyperspectral coupled with high-resolution panchromatic and/or multispectral imaging subsystems. An example is the NASA Earth Observer 1 satellite, which includes a 30 m hyperspectral sensor and a 10 m panchromatic imager. Commercial panchromatic satellite imagery approaching 1 m spatial resolution is also available. This provides the opportunity to jointly process the hyperspectral and higher resolution panchromatic imagery to potentially achieve improved detection and/or classification performance, improved change detection, and improved visual imagery for human interpretation.

A variety of techniques have been presented in the literature for merging imagery of different spatial and spectral resolution [1–15]. Many of these techniques have been designed to sharpen multispectral imagery for human interpretation using broadband panchromatic data. Component substitution methods transform the multispectral imagery and replace one component with the broadband high-resolution imagery [4,6,16]. Commonly used transformations are intensity-hue-saturation (IHS), where the intensity is replaced, or principal component analysis (PCA), where the first principal component is replaced. Clearly, information in the lower components, that may be critical in classification and detection, is not enhanced with such an approach.

High pass techniques add high spatial frequency content from the high-resolution image to the bands of the low-resolution data [3, 5, 8, 9, 16]. The technique in [1] uses a statistical approach that adds a linear combination of the high-resolution data to a pixel replicated version of the low-resolution imagery. This technique was designed to enhance the spatial resolution of the Landsat Thematic Mapper thermal band from the remaining bands. The method described in [2] and [7] models the relationship between the high-resolution image and the image to be estimated. Regression techniques are used at the available lower resolution to obtain the model parameters.

Another approach to hyperspectral resolution enhancement is based on spectral mixture analysis. A method based on the linear mixing model has been investigated that employs

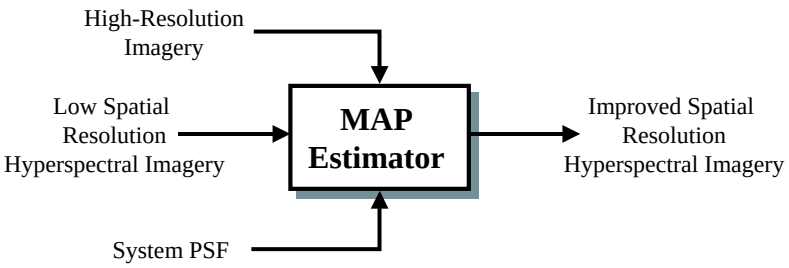


Figure 2.1: Block diagram illustrating the flow of information in the MAP estimation framework.

constrained nonlinear optimization techniques to obtain high resolution endmember fractions [12–14]. Even with the physical constraints that assure radiometric validity of the derived sub-pixel spectra, the optimization is underdetermined and not always able to arrive at a sharpened solution. An alternative approach appends the high resolution image to the hyperspectral data and computes a mixture model based on the joint data set [15]. A high resolution hyperspectral image, however, is not explicitly estimated.

Our approach is based on a novel maximum *a posteriori* (MAP) estimation framework [17] for enhancing the spatial resolution of an image using co-registered high spatial-resolution imagery from an auxiliary sensor. The estimation framework developed allows for any number of spectral bands in the primary and auxiliary sensor. The technique is suitable for applications where some correlation, either localized or global, exists between the auxiliary image and the image being enhanced. A spatially varying statistical model is used to help exploit localized correlation between the primary and auxiliary image. Another important aspect of the algorithm is that it allows for the use of an accurate observation model relating the "true" scene with the low-resolutions observations. This means that a potentially wavelength-dependent spatially-varying system point spread function (PSF) can be incorporated into the estimator. Figure 2.1 illustrates the flow of information in the estimation framework.

Figures 2.2, 2.3, 2.4, and 2.5 introduce some terminology and notation used throughout the report. Figure 2.2 shows a low resolution hypercube consisting of $M = m_v m_h$ spatial coordinates and P spectral coordinates. When represented as a one dimensional vector, a low resolution hypercube is denoted by:

$$\boldsymbol{y} = [y_{1,1}, ..., y_{P,1}, y_{1,2}, ..., y_{P,2}, ..., y_{1,M}, ..., y_{P,M}]^T. \tag{2.1}$$

Similarly, Figure 2.3 shows a multispectral data cube with $N = n_v n_h$ spatial coordinates ($N > M$) and ν spectral coordinates ($\nu < P$). For a panchromatic image, $\nu = 1$. The one

dimensional vector representing a multispectral data cube is denoted by:

$$\boldsymbol{x} = [x_{1,1}, ..., x_{\nu,1}, x_{1,2}, ..., x_{\nu,2}, ..., x_{1,N}, ..., x_{\nu,N}]^T. \tag{2.2}$$

High resolution hypercubes are denoted by $\boldsymbol{z}$ or $\hat{\boldsymbol{z}}$. As depicted in Figure 2.4, they have N spatial coordinates and P spectral coordinates. A high resolution hypercube in one dimension is denoted by:

$$\boldsymbol{z} = [z_{1,1}, ..., z_{P,1}, z_{1,2}, ..., z_{P,2}, ..., z_{1,N}, ..., z_{P,N}]^T. \tag{2.3}$$

Other terminology used throughout is shown in Figure 2.5. Note that high and low resolution hyper-pixels are one dimension vectors of length P that vary in the spectral dimension and correspond to a given spatial coordinate. A super-pixel is a collection of adjacent high resolution hyper-pixels that exactly fit into a single corresponding low resolution hyper-pixel. We require that each super-pixel correspond to a unique low resolution hyper-pixel which is equivalent to $\frac{n_v}{m_v}$ and $\frac{n_h}{m_h}$ being integers.

In Equations (2.1) and (2.3), hyper-pixels are listed in column order and similarly for Equation (2.2). For example, the first n_v hyper-pixels listed in Equation (2.3) represent the first column of hyper-pixels shown in Figure 2.4.

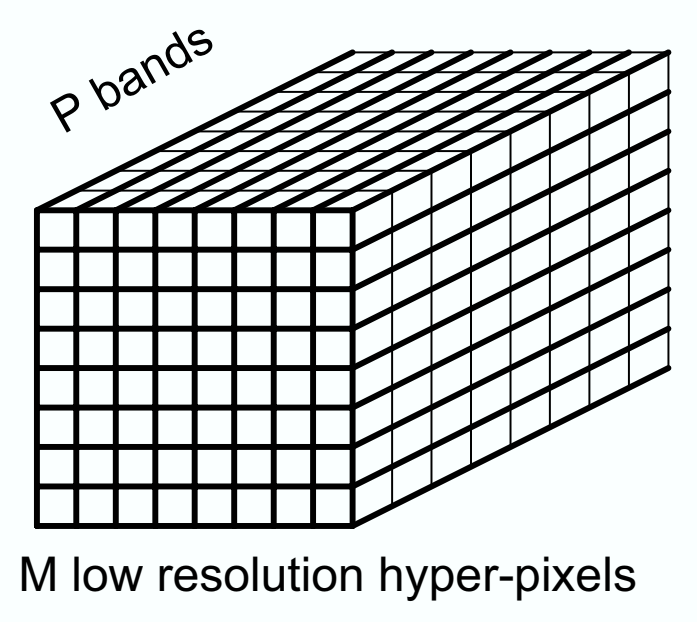


Figure 2.2: Low Resolution Hypercube $\boldsymbol{y}$

Throughout various portions of the report, we make references to Mercury computers, on which certain CPU intensive parts of the MAP algorithm were implemented. The Mercury compute system consists of multiple compute nodes interconnected by a high speed interconnect fabric called the Raceway. Each compute node consists of a high speed floating

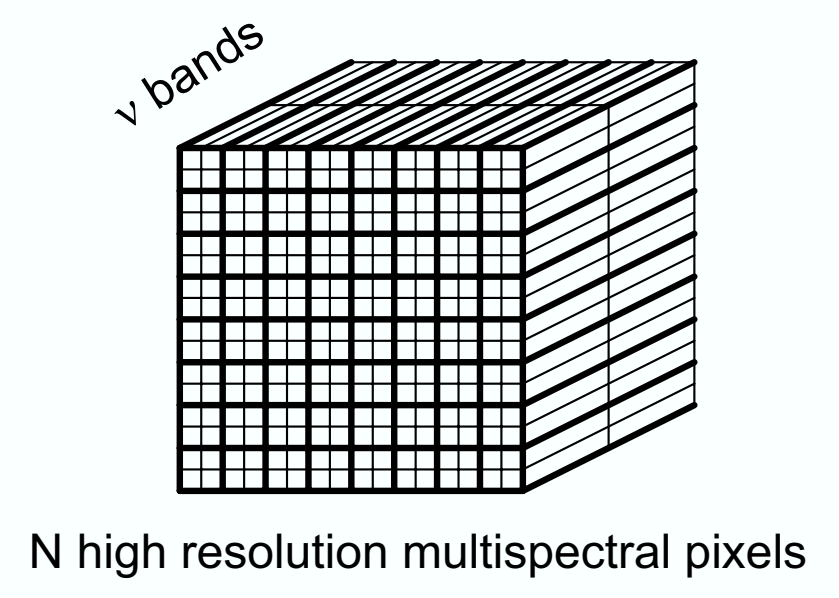


Figure 2.3: Multispectral Data Cube $\boldsymbol{x}$

point processor with an attached vector processor unit (VPU). Each node also contains local memory and an interconnection to the Raceway.

This report contains 8 chapters including this introduction. Chapter 3 discusses the delivered software, including highlighted features, installation procedure, and software use. Chapter 4 presents carefully derived explicit formulas directly implemented in the software provided. It also answers some practical questions that arose during the initial phases of implementation. Chapter 5 analyzes the performance of the algorithm and Chapter 6 presents additional analysis of the MAP algorithm with emphasis on Mercury timing. Chapter 7 discusses Stochastic Mixing Models and provides information related to interfacing them with our MAP software. Chapter 8 discusses mis-registration between input data cubes and its effect on estimation performance. Appendices A and B detail a mathematical analysis of alternate forms of the MAP algorithm, and Appendix C gives a atmospheric analysis preformed by the Nashua office of ATK Mission Research.

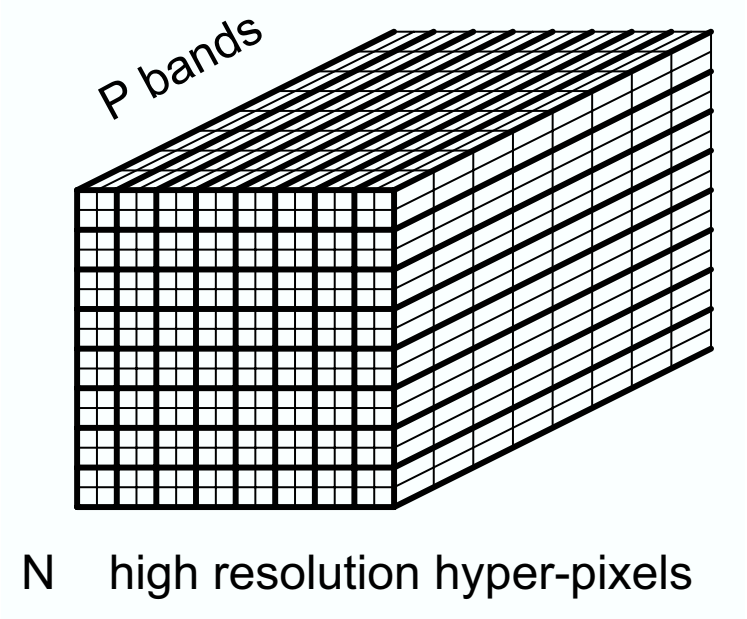


Figure 2.4: High Resolution Hypercube $\boldsymbol{z}$

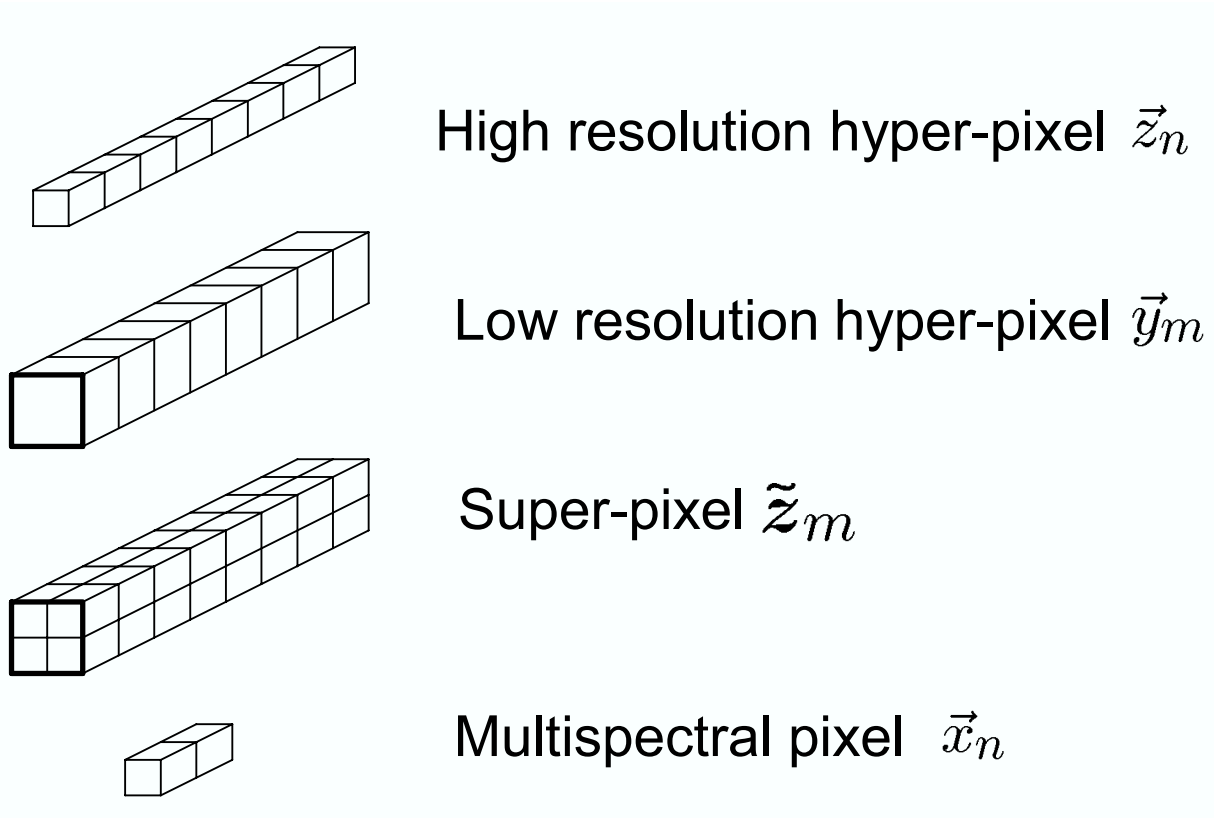


Figure 2.5: Notation and Terminology

Chapter 3

Hyperspectral Image Enhancement Software Installation and Use

A main concentration of the Phase II effort was to deliver a working implementation of the MAP algorithm on the Mercury Platform. Since one of the goals was for the delivered software to be easily accessible as a research tool, we have chosen to write all but the most CPU intensive portions in MATLAB. Secondly, due to the potentially long run times that might occur, we offload the CPU intensive parts of the implementation to a network of compute nodes running in parallel. Thirdly, because there are a number of variations of the MAP code, we incorporate them all into a single code that could be called as a function from MATLAB. Lastly, due to the complexity of inputs and calling arguments and the relationships between them, we developed a GUI interface that will reduce the time it might take for a new user to begin using the software.

We want to emphasize that we have provided a powerful tool that is a blend of code written in MATLAB (for ease of use and portability across computing architectures) and C (for speed and computing power on a Mercury system). The use of MATLAB as a user interface drastically simplifies the use of this algorithm and relieves the user from having to know details concerning the use of Mercury systems. All the user has to know is how to run MATLAB and they can tap into the power of the Mercury system.

The intensive portions of the MAP Algorithm are implemented on a Mercury computer system. However, the MATLAB scripts and GUI supplied should run on *any* computing hardware with MATLAB installed.[1] The user only need set the number of available Mercury CE's (Computational Elements) equal to 0. In the case of a Mercury computer system whose host computer has MATLAB installed, any number of available CE's may be used. Whether it's through the supplied `MAP_Algorithm.m` function or the GUI interface, the full use of all nodes on a Mercury computer has been made simple for the user.

We have tested the software on four Mercury computer systems (two with a Windows

[1]The GUI requires MATLAB version 6.5 or higher.

2000 PC host and two with a Unix Sun host) and as a result are confident that our parallel implementation is portable to any Mercury system with MATLAB installed on the host computer.[2] One such Mercury system (the AdapDev 1280) is scheduled for delivery at the end of the contract and will come with all required software pre-installed (excluding MATLAB).

As discussed below, we provide a single MATLAB function `MAP_Algorithm.m` that may be directly called by the user. Since the interface is through MATLAB, the analyst need not be concerned with the details of parallel programming, but can still benefit from the fast response times resulting from it. Other advantages of using MATLAB include plotting capabilities and a familiar computing environment.

3.1 Major Functions, Inputs, and Outputs

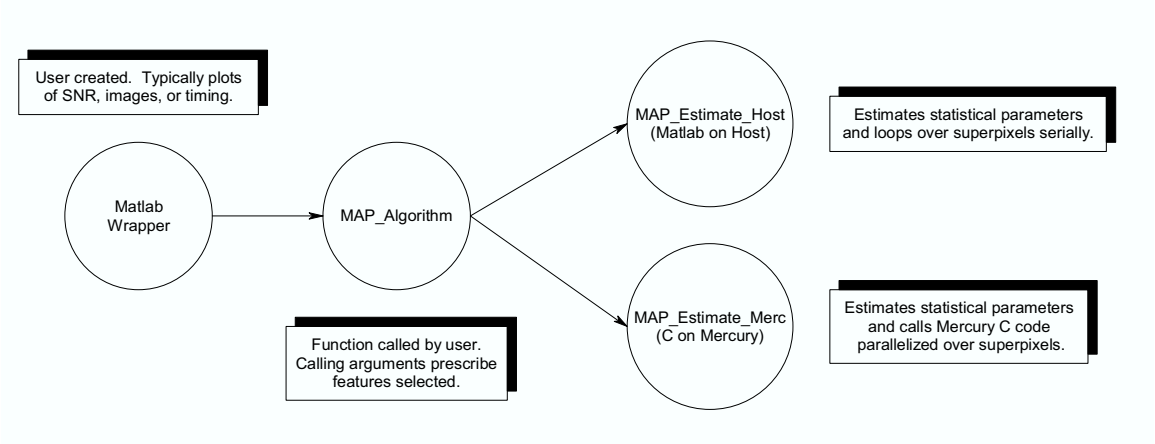


Figure 3.1: Major Functions of the MAP Algorithm

Figure 3.1 depicts the major functions of the MAP Algorithm. We see from the figure that the main function provided is `MAP_Algorithm`, which may be called directly by the user through use of a user supplied wrapper function. We have supplied several example wrapper scripts, including an easy to use GUI. The two other main functions depicted compute estimates of statistical parameters as well as the MAP estimate. The "host" function is written solely in MATLAB and is portable to any hardware platform running MATLAB. The host function makes a call to another function that computes the MAP estimate by looping over super-pixels. The "merc" function runs on a Mercury platform. It computes the statistical estimates in MATLAB and the MAP estimate through a parallel computation over super-pixels.

Figure 3.2 depicts the major inputs and outputs of `MAP_Algorithm`. Perhaps the most important inputs are a low resolution hypercube $\boldsymbol{y}$ and either a panchromatic or multispec-

[2] 1 GB or more of physical memory on the host computer is recommended.

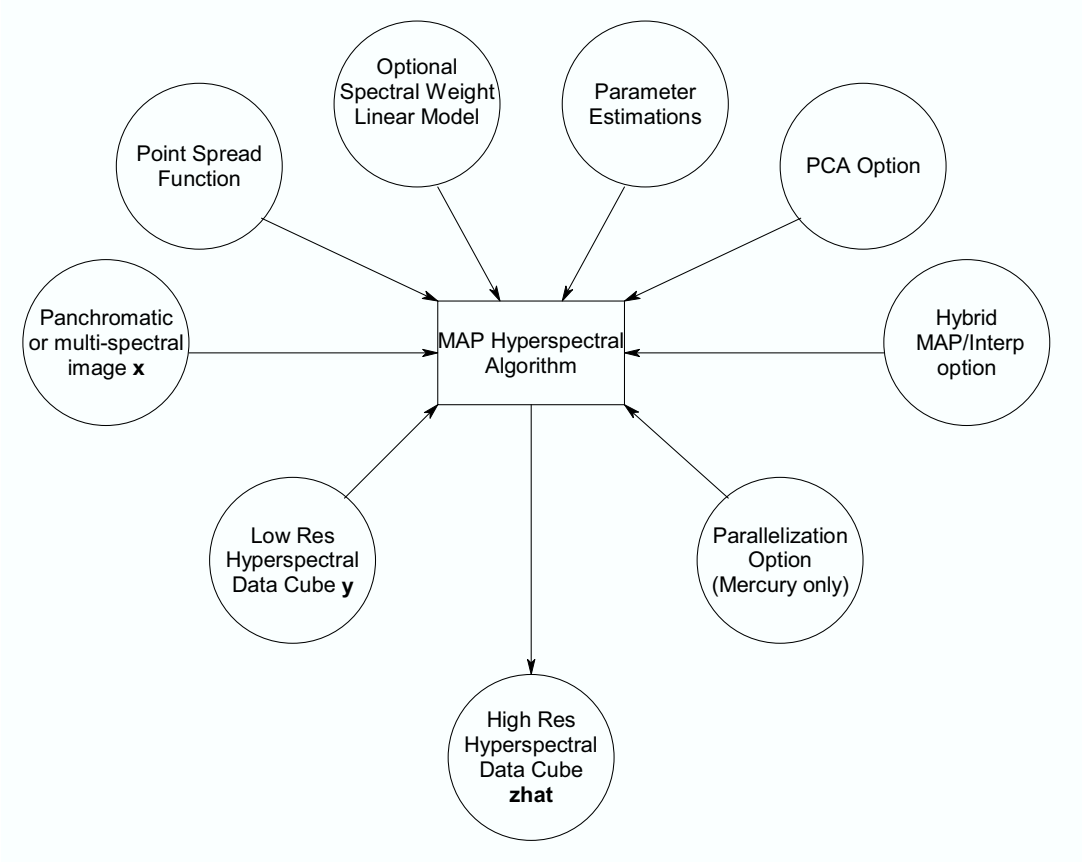


Figure 3.2: Major Inputs and Outputs of the MAP Algorithm

tral image array $\boldsymbol{x}$. These may come in the form of collected data from a two sensor platform or as degraded images from a known high resolution hypercube $\boldsymbol{z}$. The main output is the MAP estimate $\hat{\boldsymbol{z}}$ of a (possibly hypothetical) hypercube $\boldsymbol{z}$. Some of the other inputs of Figure 3.2 are briefly discussed below.

The point spread function is linear and maps a hypothetical vector $\boldsymbol{z}$ to a vector $\boldsymbol{y}$ according to the statistical model $\boldsymbol{y} = W\boldsymbol{z} + \boldsymbol{n}$. We take W to be a weighted averaging process on $\boldsymbol{z}$. The entries of each super-pixel are spatially averaged one band at a time in order to obtain the corresponding low resolution hyper-pixel. Assuming super-pixels do not overlap spatially yields a sparse matrix W with orthogonal rows and columns. It may be partitioned so that each block is a (possibly 0) multiple of an identity matrix of appropriate size. This special form of W facilitates PCA subspace analysis, because the model $\boldsymbol{y} = W\boldsymbol{z} + \boldsymbol{n}$ is preserved.

The spectral weight matrix is optional and corresponds to a similar linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$. Intuitively, this model should be assumed when frequencies corresponding to $\boldsymbol{x}$ and $\boldsymbol{y}$ overlap significantly and should not be assumed when they do not. Optionally, the matrix S may be supplied by the user or else computed from $\boldsymbol{x}$ and $\boldsymbol{y}$ using a least squares technique.

We have provided a total of four options for computing statistical parameters, three of which are self contained, and one of which requires an external Stochastic Mixing Model computation. The statistical parameter estimation plays a significant role in the determining the quality of the resulting map estimate $\hat{\boldsymbol{z}}$. Consequently, we have anticipated the desire to interface new parameter estimation algorithms and have designed the software accordingly.

There are three PCA options available. The first option is to not use PCA at all, but rather compute solely in the original "spectral components". The second option is to compute the MAP estimate in a PCA vector space and return the MAP estimate $\hat{\boldsymbol{z}}$ in the same space. The third option is a combination of the first two, namely to compute the MAP estimate in PCA space, but return the MAP estimate as spectral components. Any number of PCA components may be chosen.

Another option allows the application of a hybrid MAP-Interpolation algorithm. After specifying the number of components to be used for MAP estimation, the remaining components specified are generated using a simple interpolation method. Although the hybrid method works in both PCA space and spectral space, it is probably most useful in conjunction with PCA. Generally, there is little to gain when using the MAP estimation technique with lower ordered PCA components. When using PCA, a hybrid approach typically produces answers that compare favorably with a non-hybrid approach, but runs faster than the non-hybrid approach.

3.2 Software Highlights

As shown in Figure 3.3, `MAP_Algorithm.m` logically breaks up into 2 major components depending on whether the underlying broadband frequencies overlap or are disjoint. In the former case, a linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ between the broadband (or multispectral) image and the unknown high resolution hyperspectral image is assumed; in the later it is not. As detailed in Chapter 4 and Appendices A and B, all three forms of the MAP algorithm have been analyzed mathematically. Under the assumption of a linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$, we have proved that all three forms of our MAP estimator are identical (Section 4.5). If no such linear model is assumed, only the Form 1 MAP estimator is applicable. Therefore, the Form 1 estimator alone is sufficient for implementation, although it requires different implementations depending on whether the linear model is assumed or not.

Each implementation above breaks down further into a General Form 1 implementation and a Simplified Form 1 implementation. The latter assumes a more restricted structure of the associated covariance matrix and will run faster. Thus, there are 4 major components of the Form 1 MAP estimator that have been implemented, and all four have been integrated into the single user interface `MAP_Algorithm.m`.

Referring to Figure 3.3, the fastest run times are realized when the Simplified Form 1 Implementation is exercised. This occurs when a simplified zero-nonzero structure of the covariance matrix is assumed, and the noise parameter σ_n is zero. The simplified structure

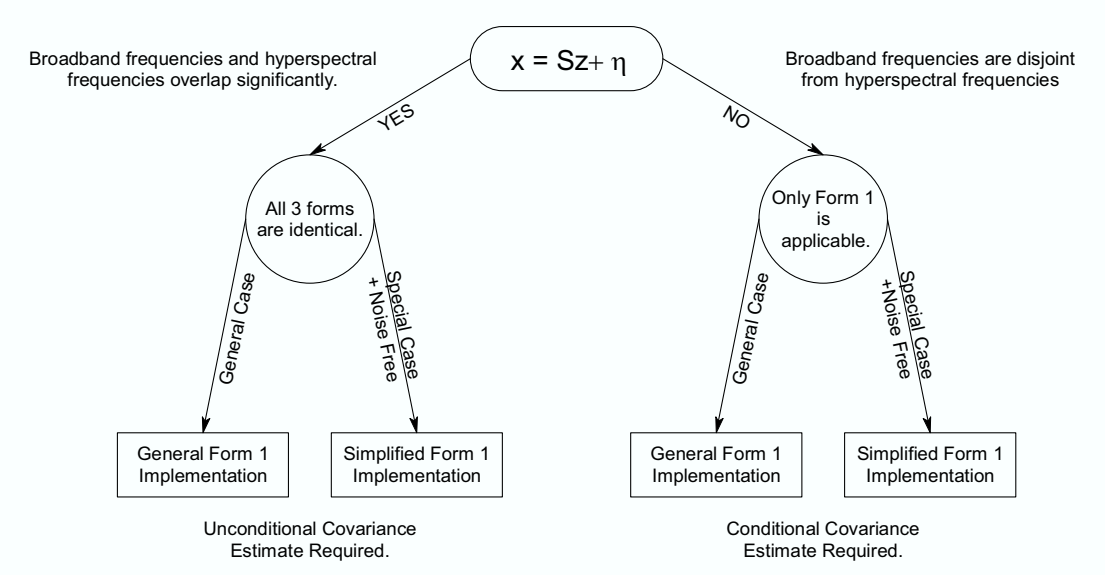


Figure 3.3: Logical Break up of the MAP Algorithm

of the covariance matrix is referred to as the "special case assumption", and the requirement that $\sigma_n = 0$ is referred to as the "noise free condition". If both the special case assumption and the noise free condition hold, a matrix inverse is avoided for each super-pixel resulting in a fast MAP estimation. Assuming a fast method of estimating covariance matrices is available, a real time Mercury implementation is feasible based on, for example, Equation (4.70).

The Simplified Form 1 implementation is automatically detected by `MAP_Algorithm.m`. All the user need do is set `var_y=0` (σ_n^2 is the input parameter `var_y` of `MAP_Algorithm.m`) and specify a covariance estimation technique consistent with the special case assumption. Speed improvements are automatically enabled when the noise free condition and special case assumptions are encountered.

In general, the code structure assumes all covariance matrices are block diagonal, where all blocks have the same size, but may otherwise be different from one another. However, the special case assumption requires additional structure. While the blocks need not be identical, they are required to be "identical within super-pixels". Details are presented in Section 4.11 of Chapter 4.

As seen in Figure 3.3, the covariance estimate required is unconditional when $\boldsymbol{x} = S\boldsymbol{z}+\boldsymbol{\eta}$ is assumed, and is conditional otherwise. In either case, a conditional covariance estimate is ultimately used as part of the Form 1 MAP Estimate. If the linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is assumed, we convert it to the conditional version using Equation (4.27). This is desirable since an unconditional covariance is easier to specify than a conditional covariance.

There are presently two ways to specify a conditional covariance matrix when $\boldsymbol{x} = S\boldsymbol{z}+\boldsymbol{\eta}$ is *not* assumed. Both are based on VQ partitioning. In one case (`ccov_meth`=1), VQ clas-

sification is based on the input hypercube $\boldsymbol{y}$ and in the other case (`ccov_meth`=2) VQ classification is based on an estimate of the conditional mean. The trade-off between these two methods is speed versus image enhancement quality. In the first method (`ccov_meth`=1), the special case assumption holds so that no matrices are inverted, and the MAP algorithm runs quickly.

There is presently one self contained algorithm for estimating the unconditional covariance matrix when $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is assumed. This is a simple estimation technique where all blocks of the block diagonal covariance matrix are identical, so that the special case assumption is met in this case as well. The Stochastic Mixing Model (SMM) interface discussed later, also falls under the assumption $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$, and it may be thought of as a second method for estimating the unconditional covariance matrix. However, this method is not self-contained since it requires input from an external SMM algorithm.

We have established a software framework for incorporating new covariance estimation techniques. A MATLAB programmer can incorporate the new estimation technique by adding a new function call to the existing MATLAB function `param_est.m`. Any new technique that adheres to the general input and output data structures established in `param_est.m` can be incorporated in a straightforward manner. We believe that these data structures are sufficiently general for the easy incorporation of new covariance estimation techniques as they become available.

The code that runs on a Mercury system makes extensive use of the Scientific Algorithm Library (SAL), which is a C library provided by Mercury. SAL is a floating point library specifically optimized for AltiVec equipped PowerPC processors. It has proven to significantly speed up codes residing on a single node. However, the functions available are somewhat limited as compared to a more mature library such as LAPACK.

We also make extensive use of another C library provided by Mercury called the Parallel Acceleration System (PAS). The PAS library is a means by which the available nodes pass data to one another. An advantage of using PAS as opposed to another communication library (e.g. DX) is its ability to scale. For example, we developed our C code on the 8 node Mercury AdapDev 1280, and then we successfully ported it to both the 96 node (Titan) and 64 node (Hawk) computer systems at Wright-Patterson Air Force Base (WPAFB). Due to the scalability of PAS, the port was accomplished with minimal changes. A disadvantage of PAS is that its portability is limited to Mercury machines, while DX is standard.

Through use of the PAS communication library, we have developed a scalable framework for projects implemented on Mercury computer systems. The intention of this framework is to allow programmers with little or no knowledge specific knowledge of Mercury computers to develop applications relatively quickly. The framework also facilitates swapping alternative functions with different argument lists as separate building blocks. As new functions are developed, swapping them with existing functions is accomplished with minimum impact to the PAS framework that surrounds them.

Other highlights include the many options available within `MAP_Algorithm.m`, some of

which are listed below:

- Work in either the spectral domain or PCA domain.
- Use MAP estimation on leading components and interpolation on the rest.
- Stochastic Mixing Model interface.
- Flexibility in specifying PSF or spectral response functions.
- Work with either panchromatic or multispectral input data.

These options are discussed further in the documentation below.

3.3 Installation

All required software has been placed in a single file. When installed (as directed below), the result is a subdirectory named `mapalgorithm`. Once you have generated this directory, all of the required files will be available to MATLAB. Windows based machines require the Winzip utility (a scaled down version of Winzip comes with Windows XP), and Linux/Unix machines require the tar utility.

For Windows based machines, unzip the compressed file `mapalgorithm.zip` to `c:\` thereby generating the directory `c:\mapalgorithm`. For Linux/Unix based machines, untar the `mapalgorithm.tar` file by issuing the command `tar xvf mapalgorithm.tar`.

In MATLAB, change the working directory to `mapalgorithm` and begin running the code using either the graphical user interface or the provided MATLAB script wrappers. For Windows based machines, the byte format option (specified within the GUI for example) should be set as little endian (LE), and for Linux/Unix based machines it should be set as big endian (BE).

3.4 Use of the GUI

Figure 3.4 displays a GUI wrapper of `MAP_Algorithm.m` named `HyperViewer.m`. It has the same functionality as the MATLAB script `MAP_Wrapper.m` described in Section 3.5, however, it is more intuitive and simpler to use. Certain options automatically become inaccessible as appropriate when other options are selected, relieving the user of remembering how the various options relate to one another.

An underlying assumption is that a known $\boldsymbol{z}$ is available to aid us in determining the performance of the MAP Algorithm. This known $\boldsymbol{z}$ will be used to generate $\boldsymbol{y}$ and $\boldsymbol{x}$. Using

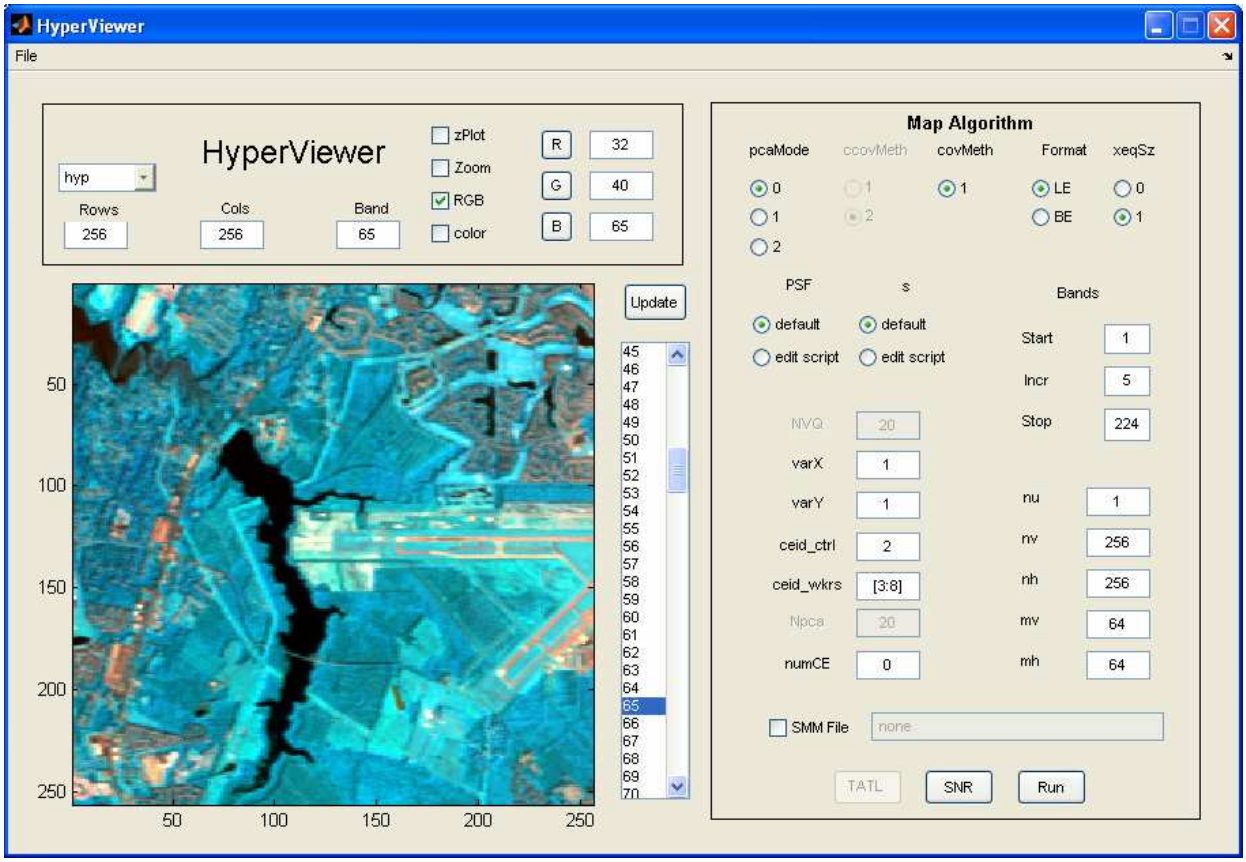


Figure 3.4: HyperViewer GUI

$\boldsymbol{x}$ and $\boldsymbol{y}$ as input, the MAP Algorithm is used to estimate a high resolution hypercube $\hat{\boldsymbol{z}}$. Then, $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$ are compared in order assess the effectiveness of `MAP_Algorithm.m`.

The starting point is a MATLAB ".mat" file containing a high resolution hypercube, which may be reduced either spatially or spectrally in order to form $\boldsymbol{z}$. This is useful if a large data set is stored, but only a subset of it is desired for analysis. The ".mat" file must contain the "full scale" hypercube as a three dimensional variable with vertical dimension first, horizontal dimension next, and spectral dimension last. An example is provided in the file aviris.mat.

After loading the ".mat" file with the MATLAB Load command, `HyperViewer` GUI is started by typing `HyperViewer(Z)` at the MATLAB prompt where `Z` is the name of the full scale hypercube stored. The variable name supplied may be anything, but it is identified in `HyperViewer` as `hyp`. In Figure 3.4 we see the full scale hypercube being displayed, indicated by the drop down window in the upper left hand corner showing the word "hyp".

Once the MAP Algorithm is run by clicking the Run button, the scaled down hypercube $\boldsymbol{z}$, the degraded images $\boldsymbol{x}$ and $\boldsymbol{y}$, and the MAP estimate $\hat{\boldsymbol{z}}$ all become available for display in the same fashion as hyp. The low resolution hypercube $\boldsymbol{y}$ is returned as two possibly different hypercubes, one (yIn) being input to `Map_Algorithm.m` and the other (yOut) being

output from `Map_Algorithm.m`. They will be different only if PCA analysis is specified by taking `pcaMode=1`.

The image corresponding to any band may be selected for viewing simply by selecting a band appearing in the scroll bar immediately to the right of the image. If RGB is checked, then three bands must be selected in order to produce an RGB image. This is done by selecting the band first within the same scroll bar, and then clicking one of the R, G, or B buttons, to assign the band to Red, Green, or Blue respectively. Alternately, these may be entered manually. Two dimensional plots of band number versus intensity may be plotted by checking "zPlot", and then clicking on the desired pixel.

The spatial dimensions of the initial scaled down hypercube $\boldsymbol{z}$ are specified as n_v and n_h for vertical and horizontal respectively. These dimensions must be less than or equal to the dimensions of the full scale hypercube hyp. The spectral dimension of $\boldsymbol{z}$ (denoted by P) is determined by specifying Start, Increment, and Stop bands of hyp. For the data specified in Figure 3.4, the bands of $\boldsymbol{z}$ will be $1, 6, 11, \ldots, 221$ so that $P = 45$. Thus, $\boldsymbol{z}$ is simply a restricted version of the full scale hypercube hyp.

The creation of $\boldsymbol{y}$ and $\boldsymbol{x}$ depend on the dimension parameters m_v, m_h, and ν (displayed as mv, mh, and nu resp.) as well as the point spread function PSF and the spectral response matrix $\boldsymbol{s}$. The high resolution $n_v \times n_h \times P$ hypercube $\boldsymbol{z}$ is degraded spatially using the PSF to obtain the $m_v \times m_h \times P$ hypercube $\boldsymbol{y}$. Similarly, $\boldsymbol{z}$ is degraded spectrally using $\boldsymbol{s}$ in order to obtain the $n_v \times n_h \times \nu$ multispectral data cube $\boldsymbol{x}$. The user has the flexibility to define his own PSF or spectral response matrix $\boldsymbol{s}$ by editing a script. Scripts yielding the default PSF and matrix $\boldsymbol{s}$ are provided as examples.

The PSF used to determine $\boldsymbol{y}$ from $\boldsymbol{z}$ should be thought of as a two-dimensional Finite Impulse Response (FIR) filter. Such a filter may be represented as an $L_v \times L_h$ matrix where $L_v = n_v/m_v$ and $L_h = n_h/m_h$. The spectral response function $\boldsymbol{s}$ is represented as a $P \times \nu$ matrix. The k^{th} column of $\boldsymbol{s}$ consists of the spectral weights used to determine the k^{th} multispectral value of $\boldsymbol{x}$ via weighted averaging of high resolution hyper-pixels. The only general requirements for $\boldsymbol{s}$ are that each entry be nonnegative and each column sum be 1. Both of these functions are discussed in detail in Chapter 4.

After selecting a PCA mode option, a covariance estimation technique, the number of VQ partitions (if applicable), and a designation of whether the linear model relating $\boldsymbol{x}$ and $\boldsymbol{z}$ is assumed, the user may run the MAP algorithm by clicking the run button. After the run is complete, an SNR plot indicating the effectiveness of `MAP_Algorithm.m` is obtained by clicking the SNR button, or a time line plot may be displayed if `numCE > 0`. Example SNR plots are presented in Chapter 5 and example time line plots are presented in Chapter 6.

After clicking the run button, all inputs and outputs of `MAP_Algorithm.m` are returned to the MATLAB base. The user may then execute any additional MATLAB functions or scripts at hand. For example, the user may wish to issue the MATLAB SAVE command in order to save all inputs and outputs of `MAP_Algorithm.m` to a ".mat" file. Since $\boldsymbol{y}$ can be

changed by MAP_Algorithm.m, input and output versions of $\boldsymbol{y}$ reside in different structures. Similarly, input and output versions of $\boldsymbol{s}$ reside in different structures. Under the file menu of the HyperViewer GUI, the user may also print a screen shot of the current state of the GUI to help document what went into generating outputs of MAP_Algorithm.m. The screen shot looks best if printed in landscape format.

Documentation for many of the options appearing in Figure 3.4 may be retrieved by resting the mouse cursor over the corresponding variable. In addition, GUI variables are a subset of the input variables of MAP_Algorithm.m, which are described in detail within Section 3.5 as well as the code comments of MAP_Algorithm.m.

Also appearing in Figure 3.4 is an option to supply a Stochastic Mixing Model (SMM) file. The format of the required file is detailed in Section 3.5, and a general description SMM is presented in Chapter 7. An SMM file named Eismann.mat is included as an example.

The byte format option little endian (LE) or big endian (BE), is dictated by whether the host computer is operating under Windows or Linux/Unix respectively. It must be set by the user as appropriate whenever a Mercury system is being used. The input cfg_file is described in the next section and also applies solely to Mercury systems. It may be set for the HyperViewer GUI by editing HyperViewer.m.

3.5 Direct Use of the MAP Algorithm Function

In addition to the HyperViewer GUI, we have created the MATLAB script MAP_Wrapper.m. Both HyperViewer and MAP_Wrapper.m may be viewed as alternative wrappers of the underlying function MAP_Algorithm.m. The wrapper MAP_Wrapper.m should be helpful as a guideline to users who wish to write their own wrapper of MAP_Algorithm.m. It is set up to make changing inputs relatively easy, and is more transparent from a programmers perspective than HyperViewer. The script is run merely by typing MAP_Wrapper at the MATLAB prompt.

The underlying assumption made by MAP_Wrapper.m is the same as that made by HyperViewer, namely that we begin with a full scale hypercube hyp and optionally reduce it to a sub-hypercube named $\boldsymbol{z}$. The inputs $\boldsymbol{y}$ and $\boldsymbol{x}$ are created by degrading $\boldsymbol{z}$ through use of the PSF and $\boldsymbol{s}$ respectively. The function MAP_Algorithm.m is then run to produce the MAP estimate $\hat{\boldsymbol{z}}$, which is compared with $\boldsymbol{z}$ in order to assess the effectiveness of MAP_Algorithm.m.

After calling the function MAP_Algorithm.m, MAP_Wrapper.m automatically plots images for bands 1,2,3, the average over all bands of $\hat{\boldsymbol{z}}$, and an SNR plot comparing $\boldsymbol{z}$ with $\hat{\boldsymbol{z}}$. If $\hat{\boldsymbol{z}}$ has been transformed into its PCA components (i.e. pca_mode==1), transforming $\boldsymbol{z}$ into its PCA components is required before a meaningful SNR plot can be made. This transformation is performed automatically by MAP_Wrapper.m as needed. Whenever one or more CEs are specified, MAP_Wrapper.m finishes by plotting a time line similar to the output

of Mercury's TATLView program. This allows for a detailed timing analysis such as that presented in Chapter 6.

Other wrappers that might be considered could involve different underlying assumptions. For example, in a realistic situation, the user would not have a high resolution hypercube (hyp) to start with. The assumption then would be that $\boldsymbol{y}$ and $\boldsymbol{x}$ are known a priori and would not be created by degrading a known $\boldsymbol{z}$. Assuming $\boldsymbol{x}$ and the hypothetical $\boldsymbol{z}$ are related through a linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$, the matrix S (created from the smaller matrix $\boldsymbol{s}$) may not be known. In this case, the user may choose to have $\boldsymbol{s}$ created by supplying it to `MAP_Algorithm.m` as an empty matrix.

Optionally in `MAP_Algorithm.m`, a hybrid MAP/interpolation may be specified where the first P_{hyb} components are computed using the MAP estimate and the last $P - P_{hyb}$ are interpolated. For simplicity, this option is not exercised in either `HyperViewer` or `MAP_Wrapper.m`, however it has been thoroughly tested and works with either spectral or PCA components. The utility of the hybrid option is that it is very fast compared to the general MAP algorithm where many $P \times P$ matrices are inverted. Of particular interest is the hybrid option in conjunction with PCA analysis, since interpolation on higher ordered components gives essentially the same SNR as the MAP estimate.

We conclude this section by describing all input and output arguments of `MAP_Algorithm.m`. As of the writing of this document, the description below matches the code comments exactly.

```
%
% Usage: mapStructOut = MAP_Algorithm (mapStructIn)
%
% The I/O arguments to map algorithm are in the form of Structures where
% the fields of the structures are given below (i.e. y = mapStructIn.y).
%
% Input (mapStructIn):
%  y:        Low resolution hypercube (mv x mh x P).
%
%  x:        Co-registered and aligned high-resolution multispectral cube
%            (nv x nh x nu).  Here aligned means that x and y span the
%            *exact* same field of view and nv/mv and nh/mh are integers.
%            Such alignment is crucial to performance.  For a panchromatic
%            image, nu = 1.
%
%  hybrid:   1 to apply MAP/interpolation hybrid, 0 for MAP alone.
%
%  P_hyb:    Number of components used in MAP portion of hybrid. The
%            remaining P - P_hyb components are 3d interpolated.
%            Requirements: 0 <= P_hyb < P  if hybrid = 1,
%                          P_hyb <= Npca   if hybrid = 1 & pca_mode > 0.
```

```
%            Set P_hyb = 0 for pure interpolation (no MAP estimation).
%
%  pca_mode: > 0 for applying MAP algorithm to principal components, 0
%            for applying MAP algorithm to original spectral components.
%            Set pca_mode=1 to return PCA coordinates and set pca_mode=2
%            to return spectral coordinates.
%
%  Npca:     Number of principal components computed if pca_mode > 0.
%            Requirements: 1 <= Npca <= P    if pca_mode > 0,
%                          P_hyb <= Npca     if hybrid = 1, pca_mode > 0.
%
%  SMM_file: A MATLAB '.mat' file containing high res abundance maps,
%            end-member mean vectors, end-member covariances, and a
%            PCA transformation matrix if pca_mode == 1.  If there is
%            no SMM file, set SMM_file = 'none'.  An SMM file is expected
%            to contain the following variables with the exact names and
%            sizes as specified below (also see MAP_SMM.m):
%                a_smm = High resolution abundance mapping (nv x nh x Ne)
%                        where Ne is the number of end members.
%                m_eps = End member mean vectors (K x Ne) where K = P if
%                        pca_mode = 0, and K = Npca otherwise.
%                C_eps = End member covariances (K x K x Ne)
%                E_smm = PCA transformation matrix (P x Npca). Take E_smm
%                        as an empty matrix if no PCA analysis was
%                        performed, in which case K = P and a_smm, m_eps,
%                        C_eps are all 'spectral domain' quantities.
%
%            Caution: If an SMM file is provided, other input parameters to
%            this function must be provided in a consistent manner.  Of
%            particular concern is the consistency of x, y, s, and nu. In
%            general, whatever (overlapping) input parameters used in the
%            creation of a_smm, m_eps, C_eps, and E_smm, should be exactly
%            the same parameters provided to this function.
%
%  Jseed:    Initial random number seed for VQ partitioning.
%
%  var_y:    Noise variance for all pixels and bands in y. In the final
%            report, this is defined as the variance of n = y - Wz.  It
%            is also a diagonal load of an inverted matrix appearing in
%            the MAP estimate.  Increasing var_y will improve the
%            conditioning of the matrix being inverted.
%
%  var_x:    Noise variance for all pixels and bands in x. In the final
```

```
%           report, this is defined as the variance of eta = x - Sz.  It
%           is also a diagonal load of an inverted matrix appearing in
%           the equation that relates conditional parameter estimates
%           (which are required for our Form 1 implementation) with the
%           corresponding unconditional parameters.  Increasing var_x
%           will improve the conditioning of the matrix being inverted.
%
%  psf:     FIR point spread function of size Lv x Lh where Lv = nv/mv,
%           and Lh = nh/mh.
%
%  As described in the final report, specification of a psf matrix of size
%  Lv x Lh is equivalent to taking all rows of the M x L matrix omega_hat
%  to be psf(:)' where M = mv*mh and L = Lv*Lh.  As presented in the
%  report, omega_hat contains the non-zero entries of the M x N matrix
%  omega and the MP x NP matrix W is given in terms of omega by:
%  W = Q*kron(omega,eye(P)).
%
%  xeqSz:     1 if x and z are linearly related by x = Sz + eta.
%             0 otherwise.
%
%  cov_meth:  Method of estimating the (unconditional) covariance matrix Cz
%             This method applies only when xeqSz=1. Presently one type:
%               cov_meth = 1: all PxP blocks are identical.
%
%  ccov_meth: Method of estimating the conditional covariance matrix Cz|x
%             This method applies only when xeqSz = 0. Presently two types:
%               ccov_meth = 1: VQ partitioning, classification on y.
%               ccov_meth = 2: VQ partitioning, classification on Muz|x.
%
%  I_meth:    Method of interpolation for obtaining mean images from
%             down sampled images.  Same options as interp2 (e.g.'linear').
%
%  NVQ:       Number of partitions in VQ (if applicable).
%
%  s:         Spectral response matrix if applicable (i.e. if xeqSz = 1).
%             Specify as an empty matrix in order to compute it using
%             a constrained least squares technique.  If nonempty, the
%             size of s should be P x nu unless both hybrid=1 and
%             pca_mode=0, in which case the size should be P_hyb x nu.
%             Each entry of s must be nonnegative and each column must
%             sum to unity.
%
%             The P x nu spectral response matrix s (if provided at all)
```

```
%             should be provided in the spectral domain.  If PCA analysis
%             is to be performed (i.e. pca_mode > 0), s will be
%             automatically transformed into PCA space.  Therefore, if
%             pca_mode > 0, s will be transformed by s = Ey'*s, where Ey
%             is the P x Npca matrix described below.  The resulting size
%             of s is Npca x nu.
%
%             If hybrid = 1 and pca_mode > 0, the Npca x nu matrix s is
%             later restricted to its first P_hyb components.  Thus,
%             regardless of the value of pca_mode, the final size of
%             s is  P_hyb x nu  whenever  xeqSz = hybrid = 1.
%
% num_CE:     Number of nodes (CE's) to use on a Mercury computer system.
%             Set this input to 0 if not running on a Mercury.  Otherwise,
%             setting this to 0 will run MAP_Algorithm on whatever host
%             computer is being used as interface to the Mercury.
%
% Input required if running on a Mercury computer system:
%   WorkerStack:  size of the stack on each worker CE (# bytes)
%   WorkerHeap:   size of the heap on each worker CE (# bytes)
%   ControlStack: size of the stack on the controller CE (# bytes)
%   ControlHeap:  size of the heap on the controller CE (# bytes)
%       The controller heap size may need to increase when the controller
%       is the only node specified for execution.  It may be decreased
%       proportionally as worker nodes are specified, because each node
%       will need proportionally less memory to hold its assigned
%       super-pixels.  Page faults and resources exhausted errors can
%       many times be corrected by adjusting the controller heap size.
%   ceid_wkrs:    array of worker nodes to run on the Mercury system
%   ceid_ctrl:    the CE id of the control node on the Mercury system
%   byte_format:  Byte format of host: 1 = little endian (e.g. Intel host)
%                                      2 = big endian (e.g. Sun host)
%   cfg_file: Name and location of Mercury '.cfg' file for application
%             of a configmc command on the Mercury host.  For example,
%               cfg_file = 'C:\MercurySoftware\AdapDev\AdapDev.cfg'
%             will issue the following command from MAP_estimate_merc.m:
%               configmc -cf C:\MercurySoftware\AdapDev\AdapDev.cfg init
%             The intent is to have an automatic way of avoiding "out of
%             resources" crashes on the Mercury. The variable cfg_file is
%             ignored if not running on a Mercury (designated by taking
%             numCE = 0).  To avoid the application of a configmc command
%             when running on a Mercury, set cfg_file = 'skip'.
%
```

```
%
% Output (mapStructOut):
%
%   Pzhat:    Size of the spectral dimension of the MAP estimate. It is
%             identical to the input P (i.e. third dim of input y) unless
%             pca_mode = 1.  If pca_mode = 1, then Pzhat = Npca.
%
%   y:        Low resolution hypercube of size mv x mh x Pzhat.  If
%             pca_mode = 1, y is transformed into PCA space.  Otherwise, it
%             is identical to the input y.
%
%   s:        Spectral response matrix (if applicable) adjusted according to
%             a principal component transformation if pca_mode = 1.  The
%             output size is  Pzhat x nu  unless hybrid = 1, in which
%             case the output size is P_hyb x nu.
%
%   zhat:     The MAP estimate: a high resolution hypercube of size
%             nv x nh x Pzhat.  If pca_mode = 1, principal components are
%             returned.  If pca_mode = 2, processing is performed in PCA
%             coordinates but the resulting MAP estimate is returned in the
%             original spectral coordinates.
%
%   Cy:       P x P spectral covariance matrix used for PCA transformation.
%             Returned as an empty matrix if pca_mode = 0. Here, P is the
%             size of the spectral dimension of the input y.
%
%   Ey:       P x Npca  matrix of eigenvectors of Cy. The jth column of Ey
%             is the eigenvector corresponding to the jth eigenvalue of Cy
%             where the Npca eigenvalues sorted from largest to smallest.
%             Returned as an empty matrix if pca_mode = 0.
%
```

3.6 Other Wrappers

Also provided is a third wrapper of `MAP_Algorithm.m`, named `MAP_Wrapper_diagnostic.m`, whose purpose is to provide the user with diagnostic information about the magnitude of differences between running on the host and running on the network of CEs. An important application of this is to identify malfunctioning nodes. We have used the diagnostic wrapper to identify a bad node (node 77 as determined by the default configuration file) on the Titan 96 node Mercury system at WPAFB. The cause is likely to be faulty local memory at that node.

Code run on the network of CEs is written in single precision C. Conversely, code run on the host computer is written in double precision MATLAB. Due to the different precisions involved, there will be differences between the resulting $\hat{\boldsymbol{z}}$ that each produces. In order to quantify these differences, `MAP_Wrapper_diagnostic.m` calls the `MAP_Algorithm` twice in succession: first on the host computer, and the next on the network of CEs. Afterward, a plot is produced, showing the magnitude of the (mean) differences on the vertical axis and the CE identification on the horizontal axis.

Editing the script `MAP_Wrapper_diagnostic.m`, the user may specify any desired input parameters. The value of `mapStructIn.num_ce` is automatically set to zero for the first call to `MAP_Algorithm.m`, and automatically set to the user specified value of the variable `num_ce` for the second call. After the desired inputs are supplied, the diagnostic wrapper is run as a MATLAB script simply be typing `MAP_Wrapper_diagnostic.m` at the MATLAB prompt.

Figure 3.5 shows an example of running `MAP_Wrapper_diagnostic.m` on the Mercury AdapDev 1280. Each marker on the plot represents the difference in $\hat{\boldsymbol{z}}$ corresponding to a particular super-pixel. The CE that processed the super-pixel lies directly underneath the marker. We see in Figure 3.5 that all differences between the two runs are relatively small, indicating that both implementations agree with one another and that all nodes tested are functioning properly. An unusually large difference at a particular CE would indicate a problem at that CE.

A fourth wrapper of `MAP_Algorithm.m`, named `plot_timing.m`, generates timing plots as a function of available nodes. In this wrapper, several runs of `MAP_Algorithm.m` are made is succession, where the number of nodes used for a given run varies while the other input variables are fixed. These plots help access the utility of the Mercury platform as it pertains to this project. Several examples are presented in Chapter 6.

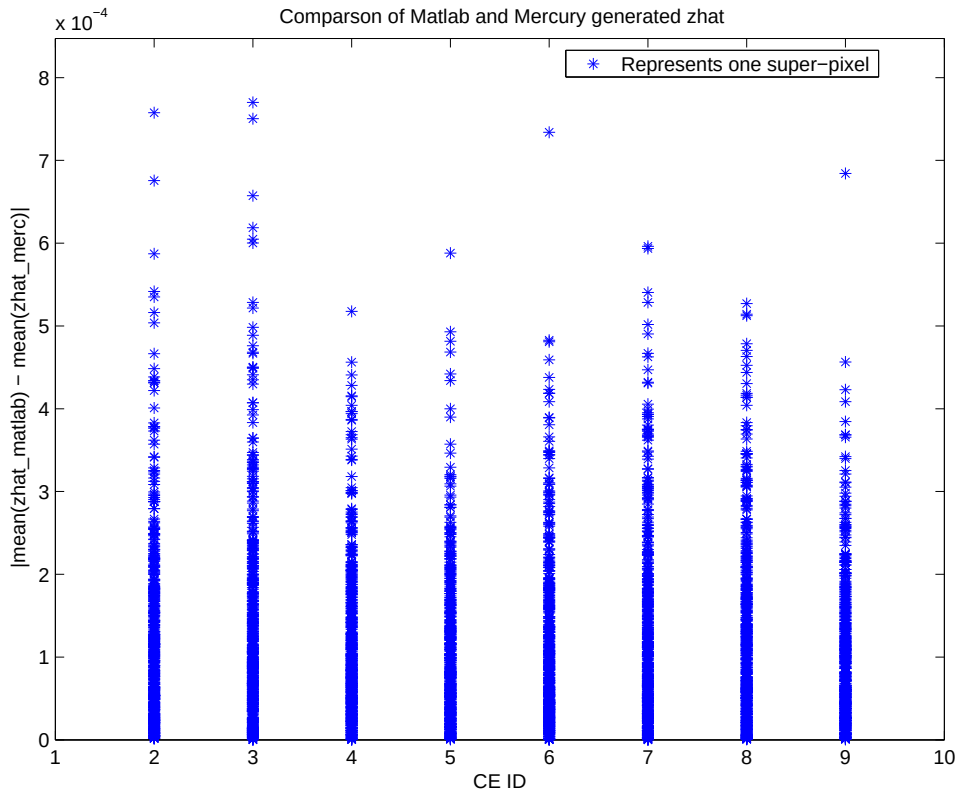


Figure 3.5: Diagnostics on the AdapDev 1280

Chapter 4

Mathematical Derivations and Analysis

This chapter presents the mathematical framework relating to the theory and computer implementation of a Maximum A Posteriori (MAP) algorithm for image enhancement. A number of practical questions that arose during the initial phases of implementation are addressed and answered here. Some of the topics addressed are the singular or non-singular status of certain matrices that require inversion, the implication of making various covariance matrices diagonal, the relationship between conditional independence and diagonal covariance matrices, and the conditions under which the various forms of the MAP algorithm are identical. Careful attention has been paid to accurately detail the mathematical equations required for implementation on either a serial or parallel computer system.

Almost universally, we assume the same notation that appears in the article [18]. In particular, a broadband image array $\boldsymbol{x}$ (which is generalized to a multispectral image in Sections 4.14 and 4.13), a low resolution hypercube $\boldsymbol{y}$, and a high resolution hypercube $\boldsymbol{z}$ have entries in band-sequential lexicographical order:

$$\begin{aligned}
\boldsymbol{x} &= (x_1, x_2, \ldots, x_N)^T, \\
\boldsymbol{y} &= (y_{1,1}, y_{2,1}, \ldots, y_{P,1}, y_{1,2}, y_{2,2}, \ldots, y_{P,2}, \ldots, y_{1,M}, y_{2,M}, \ldots, y_{P,M})^T, \\
\boldsymbol{z} &= (z_{1,1}, z_{2,1}, \ldots, z_{P,1}, z_{1,2}, z_{2,2}, \ldots, z_{P,2}, \ldots, z_{1,N}, z_{2,N}, \ldots, z_{P,N})^T.
\end{aligned}$$

Thus, a broadband image array consists of N pixels, a high resolution hypercube consists of P bands with N pixels in each band, and a low resolution hypercube has P bands with $M < N$ pixels in each band. The m^{th} low resolution hyper-pixel is given by $\vec{y}_m = (y_{1,m}, y_{2,m}, \ldots, y_{P,m})^T$, and the i^{th} high resolution hyper-pixel is given by $\vec{z}_i = (z_{1,i}, z_{2,i}, \ldots, z_{P,i})^T$. A super-pixel is a collection of adjacent high resolution hyper-pixels and are associated with a given low resolution hyper-pixel as described in Section 4.2.

As seen in the above ordering for $\boldsymbol{y}$ and $\boldsymbol{z}$, the spectral dimension of length P is listed first. We denote the vertical and horizontal spatial dimensions of $\boldsymbol{y}$ by m_v and m_h and

those of $\boldsymbol{z}$ by n_v and n_h so that $M = m_v m_h$ and $N = n_v n_h$. Referring to Figure 4.1 (where $n_v = 4$, $n_h = 6$, $m_v = 2$, and $m_h = 3$), we see that the high resolution hyper-pixel $\vec{z}_i$ lying in row k and column j has index $i = (j-1)n_v + k$ for $1 \leq k \leq n_v$ and $1 \leq j \leq n_h$. Similarly, the low resolution hyper-pixel $\vec{y}_m$ lying in row k and column j has index $m = (j-1)m_v + k$ for $1 \leq k \leq m_v$ and $1 \leq j \leq m_h$. Therefore, hyper-pixels $\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_M$ and $\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_N$ are listed in column order. The entries of the broadband image array $\boldsymbol{x}$ are listed in column order as well.

We assume that $\boldsymbol{z}$ is a multivariate random variable with a prior Gaussian distribution and further assume Bayesian general linear models (see 10.6 of [17]):

$$\begin{aligned} \boldsymbol{x} &= S\boldsymbol{z} + \boldsymbol{\eta}, \\ \boldsymbol{y} &= W\boldsymbol{z} + \mathbf{n}, \end{aligned}$$

where S denotes an $N \times NP$ spectral response matrix[1], and W denotes the $MP \times NP$ matrix form of a point spread function. By definition of a Bayesian general liner model, the vectors $\boldsymbol{\eta}$ and $\mathbf{n}$ are zero mean multivariate Gaussian random variables and are independent of $\boldsymbol{z}$. It follows from these assumptions that $\boldsymbol{x}$ and $\boldsymbol{z}$ are jointly Gaussian as are $\boldsymbol{y}$ and $\boldsymbol{z}$.

The model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is only assumed optionally, since we also want to examine the case where it may not hold. The model would most naturally be assumed when the frequency band associated with the known hypercube $\boldsymbol{y}$ contains the frequency band associated with the panchromatic image $\boldsymbol{x}$. The model would most likely not be assumed if these frequency bands are disjoint. Much of this chapter applies regardless of whether or not the model is assumed. The way to decern if an equation depends on the model is by simply observing whether the matrix S (or its related quantity $\mathbf{s}$) appears.

Although some results are presented in terms of arbitrary covariance matrices C_η and C_n of $\boldsymbol{\eta}$ and $\mathbf{n}$, we will typically make the additional assumption that they are multiples of the appropriate identity matrix:

$$\begin{aligned} C_\eta &= \sigma_\eta^2 I_N, \\ C_n &= \sigma_n^2 I_{MP}. \end{aligned}$$

There are three equivalent forms of MAP estimators address here. Each form is represented in terms of a product of probability density functions that is to be maximized over all possible $\boldsymbol{z}$. The argument $\boldsymbol{z}$ for which maximum is attained, is expressed as the solution to a matrix equation. The derivation of each such matrix equation from its PDF representation, and a proof of their equivalence, is presented in Section 4.5.

[1]The way S is defined later in Equation (4.43), $\boldsymbol{x} - \boldsymbol{\eta} = S\boldsymbol{z}$ will provide $\boldsymbol{x} - \boldsymbol{\eta}$ in super-pixel ordering (defined later), assuming $\boldsymbol{z}$ is given lexicographically.

4.1 Kronecker Products

Since significant use is made of the direct sum and the Kronecker tensor product [19], we define them below and list some of their properties. The direct sum of any matrices A and B is given by:

$$A \bigoplus B \;=\; \mathrm{diag}(A, B) \;=\; \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}.$$

Extending this in the obvious way to a direct sum of M matrices we have:

$$\begin{aligned} \left(\bigoplus_{m=1}^{M} A_m\right)\left(\bigoplus_{m=1}^{M} B_m\right) &= \bigoplus_{m=1}^{M} A_m B_m, \\ \left(\bigoplus_{m=1}^{M} A_m\right)^T &= \bigoplus_{m=1}^{M} A_m^T, \\ \left(\bigoplus_{m=1}^{M} A_m\right)^{-1} &= \bigoplus_{m=1}^{M} A_m^{-1}, \end{aligned}$$

where A_m and B_m are assumed to have compatible sizes for matrix multiplication.

For any matrices A and B, the (right) Kronecker tensor product of A and B is defined by:

$$A \otimes B \;=\; \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n}B \\ \vdots & \vdots & & \vdots \\ a_{m,1}B & a_{m,2}B & \dots & a_{m,n}B \end{bmatrix},$$

where $a_{i,j}$ denotes the (i, j) entry of the $m \times n$ matrix A. The Kronecker product is an intrinsic MATLAB function "kron" where:

$$A \otimes B \;=\; \mathrm{kron}(A, B).$$

Several useful facts concerning Kronecker tensor products are:

$$\begin{aligned} (A \otimes B) \otimes C &= A \otimes (B \otimes C), && (4.1) \\ A \otimes (B + C) &= A \otimes B + A \otimes C, && (4.2) \\ (B + C) \otimes A &= B \otimes A + C \otimes A, && (4.3) \\ (A \otimes B)^T &= A^T \otimes B^T, && (4.4) \\ (A_1 \otimes B_1)(A_2 \otimes B_2) &= A_1 A_2 \otimes B_1 B_2, && (4.5) \end{aligned}$$

where the matrices appearing within ordinary matrix operations have compatible sizes. The additional property $(A \otimes v)B = AB \otimes v$ where v is a column or row vector, may be derived from (4.5) by setting $B = B \otimes 1$, and similarly for the property $(v \otimes A)B = v \otimes AB$.

4.2 Super-pixel Ordering

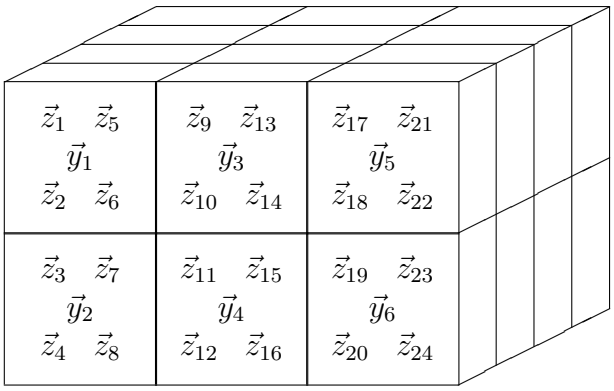


Figure 4.1: Hypercube partitioned into super-pixels

Figure 4.1 depicts a high resolution hypercube with $N = 24$, $M = 6$, and $P = 4$ partitioned into super-pixels. The column ordering of the high resolution hyper-pixels is $\boldsymbol{z} = (\vec{z}_1; \vec{z}_2; \ldots; \vec{z}_{24})$, where the semicolon denotes vertical concatenation. The above partitioning defines a new ordering given by listing super-pixels in column order. Defining q to be the permutation

$$q = \begin{pmatrix} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21\ 22\ 23\ 24 \\ 1\ 2\ 5\ 6\ 3\ 4\ 7\ 8\ 9\ 10\ 13\ 14\ 11\ 12\ 15\ 16\ 17\ 18\ 21\ 22\ 19\ 20\ 23\ 24 \end{pmatrix},$$

the super-pixel ordering of $\boldsymbol{z}$ is given by $\tilde{\boldsymbol{z}} = (\vec{z}_{q(1)}; \vec{z}_{q(2)}; \ldots; \vec{z}_{q(24)})$. Note that the ordering of $\boldsymbol{y}$ is unaffected since super-pixels are listed in column order. These remarks extend to the general case where $N = n_h n_v$ and $M = m_h m_v$ provided that $\frac{n_v}{m_v}$ and $\frac{n_h}{m_h}$ are integers. In this case, each super-pixel is a $\frac{n_v}{m_v} \times \frac{n_h}{m_h} \times P$ hypercube and the hyper-pixels of adjacent super-pixels do not overlap. In the above example $n_v = 4$, $n_h = 6$, $m_v = 2$, and $m_h = 3$. The subscripts h and v are used to represent horizontal and vertical directions of a hypercube.

The $NP \times NP$ permutation matrix corresponding to q is obtained by partitioning the $NP \times NP$ identity matrix into $P \times P$ blocks and for $1 \leq j \leq N$, placing its j^{th} block column into block column number $q(j)$. Denoting the resulting permutation matrix by Q, the high resolution hypercube in the super-pixel ordering is simply $\tilde{\boldsymbol{z}} = Q\boldsymbol{z}$. Since Q is an orthogonal matrix, $Q^{-1} = Q^T$ will always hold. In the above example, we have in addition $Q = Q^T$, however, this is not true in general (the case $(n_v, n_h, m_v, m_h) = (9, 2, 3, 1)$ is a counterexample). For programming efficiency, any matrix operation involving Q may be replaced by a corresponding operation with q.

4.3 Point Spread Functions

The matrix W appearing in the Bayesian general linear model $\boldsymbol{y} = W\boldsymbol{z} + \mathbf{n}$ represents the Point Spread Function (PSF). It is an $MP \times NP$ matrix that when applied to $\boldsymbol{z}$ performs blurring and down-sampling. For the purposes of discussion and introduction, we assume for the moment that W represents an un-weighted averaging transformation: the high resolution hyper-pixels belonging to a super-pixel are averaged to obtain the corresponding low resolution hyper-pixel. This will subsequently be generalized to a weighted average. In terms of the permuted high resolution hyper-pixel $\tilde{\boldsymbol{z}}$, the linear model is $\boldsymbol{y} = WQ^T\tilde{\boldsymbol{z}} + \mathbf{n}$, or equivalently $\boldsymbol{y} = \widetilde{W}\tilde{\boldsymbol{z}} + \mathbf{n}$ where $\widetilde{W} = WQ^T$.

Since both W and $\widetilde{W}$ are $MP \times NP$ matrices, they may partitioned into $P \times P$ blocks where each block acts on a high resolution hyper-pixel. Since W represents an averaging of super-pixels and there are N/M hyper-pixels per super-pixel, each block is either the zero matrix or else is $\frac{M}{N}I_P$. The nonzero blocks of $\widetilde{W}$ have a simpler form than those of W. For the previous example of $N = 24$, $M = 6$, and $P = 4$, the matrix $\widetilde{W}$ has the form:

$$\widetilde{W} = \begin{bmatrix} * & * & * & * & & & & & & & & & & & & & & & & & & & & \\ & & & & * & * & * & * & & & & & & & & & & & & & & & & \\ & & & & & & & & * & * & * & * & & & & & & & & & & & & \\ & & & & & & & & & & & & * & * & * & * & & & & & & & & \\ & & & & & & & & & & & & & & & & * & * & * & * & & & & \\ & & & & & & & & & & & & & & & & & & & & * & * & * & * \end{bmatrix}, \tag{4.6}$$

where each $*$ represents $\frac{1}{4}I_4$.

This simple form is preserved whenever super-pixels are non-overlapping. When this occurs, each row of $\widetilde{W}$ consists of $\frac{N}{M}$ nonzero $P \times P$ blocks $\frac{M}{N}I_P$ placed diagonally as in Equation (4.6). Now let $L = \frac{N}{M}$ and define W_L to be the $P \times LP$ matrix consisting of L copies of $\frac{1}{L}I_P$ concatenated horizontally. Then, W_L may be succinctly denoted by:

$$W_L = \frac{1}{L}J_{1\times L} \otimes I_P, \tag{4.7}$$

where $J_{1\times L}$ is the $1 \times L$ vector of all ones and $\otimes$ is the Kronecker tensor product. Likewise, the point spread function $\widetilde{W}$ is:

$$\widetilde{W} = I_M \otimes W_L = \bigoplus_{m=1}^{M} W_L. \tag{4.8}$$

Thus, $\widetilde{W}$ consists of M non square blocks on the diagonal (each of size $P \times LP$) resulting in a matrix of size $MP \times NP$. We will refer to this example as the un-weighted averaging PSF.

We now relax the assumption that $\widetilde{W}$ has the form (4.8) but still insist on:

1. Super-pixels do not overlap.

2. The PSF does not vary spectrally.

As we have seen, the first assumption permits the use of a permutation q to describe the super-pixel ordering of high resolution hyper-pixels. The second assumption means that the PSF is fully prescribed by an arbitrary $M \times N$ matrix ω. If we fix a specific spectral value λ from among the P available, and consider the N vector z_λ and M vector y_λ corresponding to it, then $y_\lambda = \omega z_\lambda + n_\lambda$ where n_λ is the associated noise. The meaning of the second assumption is that ω does not depend on λ.

For general ω, the PSF matrix $\widetilde{W}$ is given by:

$$\widetilde{W} \;=\; \omega \otimes I_P.$$

In particular, from Equations (4.7) and (4.8),

$$\begin{aligned} \widetilde{W} &= I_M \otimes \left(\frac{1}{L} J_{1\times L} \otimes I_P\right), \\ &= \left(\frac{1}{L} I_M \otimes J_{1\times L}\right) \otimes I_P, \end{aligned}$$

so that the $M \times N$ matrix ω of the un-weighted averaging example is given by:

$$\omega \;=\; \frac{1}{L} I_M \otimes J_{1\times L}. \tag{4.9}$$

To see how the PSF not varying spectrally affects $\widetilde{W}$, we partition it into $P \times P$ blocks $\widetilde{W}_{i,j}$ where $1 \le i \le M$ and $1 \le j \le N$. The second assumption is equivalent to:

$$\widetilde{W}_{i,j} = \omega_{i,j} I_P \quad (1 \le i \le M, \quad 1 \le j \le N),$$

where $\omega_{i,j}$ is the (i,j) entry of ω and I_P is the $P \times P$ identity matrix. This simple structure of $\widetilde{W}_{i,j}$ will be important when the Form 1 MAP estimate is considered in conjunction with principle component analysis.

If we make no additional assumptions regarding $\widetilde{W}$, the matrix inversions appearing in the MAP estimates are potentially full $MP \times MP$ matrices, even when the associated covariance matrices have simple structures such as being block diagonal with $P \times P$ blocks. In order to avoid working with full $MP \times MP$ matrices, we are led to consider a third assumption:

3. The PSF combines only the high resolution hyper-pixels making up a super-pixel in order to form the corresponding low resolution hyper-pixel.

This is equivalent to assuming that $\widetilde{W}$ is an appropriately generalized version of the form (4.6), namely:

$$\widetilde{W} = \bigoplus_{m=1}^{M} W_m, \tag{4.10}$$

where W_m is the $P \times LP$ matrix given by:

$$W_m = (\omega_{m,(m-1)L+1} I_P,\, \omega_{m,(m-1)L+2} I_P,\, \ldots,\, \omega_{m,mL} I_P), \quad 1 \le m \le M. \tag{4.11}$$

Here, the matrices W_m play the role of the matrix W_L introduced previously. The third assumption is characterized by the condition that ω is block diagonal with $1 \times L$ blocks:

$$\omega_{m,k} \neq 0 \quad \text{only for} \quad (m-1)L + 1 \le k \le mL \qquad (1 \le m \le M),$$

or equivalently:

$$\omega \;=\; \bigoplus_{m=1}^{M} \vec{\omega}_m^T$$

where

$$\vec{\omega}_m \;=\; (\omega_{m,(m-1)L+1}, \omega_{m,(m-1)L+2}, \ldots, \omega_{m,mL})^T.$$

To simplify the notation, define $\hat{\omega}_{m,j} = \omega_{m,(m-1)L+j}$ so that $\hat{\omega}$ is $M \times L$ and

$$\vec{\omega}_m \;=\; (\hat{\omega}_{m,1}, \hat{\omega}_{m,2}, \ldots, \hat{\omega}_{m,L})^T,$$

making $\vec{\omega}_m^T$ the m^{th} row of $\hat{\omega}$.

For the un-weighted averaging PSF given by Equation (4.9), $\hat{\omega}_{m,j} = \frac{1}{L}$ for $1 \le m \le M$ and $1 \le j \le L$. For a weighted averaging PSF, we may define the $M \times L$ matrix $\hat{\omega}$ in any fashion.

An alternate notation for W_m is:

$$W_m \;=\; \vec{\omega}_m^T \otimes I_P,$$

so that

$$\widetilde{W} \;=\; \bigoplus_{m=1}^{M} \left(\vec{\omega}_m^T \otimes I_P\right). \tag{4.12}$$

Conditions 1, 2 & 3 are summarized by the assertion that each entry of $\boldsymbol{y}$ corresponds to a unique band/super-pixel pair and is determined from $\boldsymbol{y} = W\boldsymbol{z} + \mathbf{n}$ as a weighted average of only those entries of $\boldsymbol{z}$ that lie in the corresponding band and super-pixel.

Throughout the remainder of this chapter, we assume all three of the above conditions on the PSF. If they don't hold in practice, it will be desirable to approximate the true PSF with one where they do hold.

In our MATLAB implementation of the MAP algorithm, we have chosen to think of a PSF as a Finite Impulse Response (FIR) filter. Such a filter may be represented as an $L_v \times L_h$ matrix where $L_v = n_v/m_v$, and $L_h = n_h/m_h$ (so $L = L_v L_h$). In the present context, this is equivalent to taking all rows of the $M \times L$ matrix $\hat{\omega}$ to be identical, where each row consisting of the entries of the $L_v \times L_h$ FIR matrix representation listed in column order.

4.4 Three Forms of MAP Estimators

The maximum a posteriori (MAP) estimate ([17], p.350) of $\boldsymbol{z}$ given $\boldsymbol{y}$ and $\boldsymbol{x}$ is defined by:

$$\hat{\boldsymbol{z}} \;=\; \arg\max_{\boldsymbol{z}} P_r\left(\boldsymbol{z}|\boldsymbol{\psi}\right), \tag{4.13}$$

where

$$\boldsymbol{\psi} \;=\; \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix},$$

and P_r denotes the probability density function of the continuous random variable $\boldsymbol{z}|\boldsymbol{\psi}$. We derive three alternate forms of this estimate, each corresponding to it own function of $\boldsymbol{z}$ to be maximized. It will be seen in Section 4.5 that all three are equal to $E(\boldsymbol{z}|\boldsymbol{\psi})$.

The three forms are most easily derived in reverse order as follows. From Bayes rule:

$$P_r\left(\boldsymbol{z}|\boldsymbol{\psi}\right) \;=\; \frac{P_r\left(\boldsymbol{\psi}|\boldsymbol{z}\right) P_r(\boldsymbol{z})}{P_r\left(\boldsymbol{\psi}\right)}, \tag{4.14}$$

where we have chosen not give different names to different density functions since the function argument will make it clear which PDF is being referred to. Since $P_r\left(\boldsymbol{\psi}\right)$ is not a function of $\boldsymbol{z}$, the MAP estimate (4.13) is unaffected if it is ignored, thus giving the Form 3 MAP estimate:

$$\hat{\boldsymbol{z}}_3 \;=\; \arg\max_{\boldsymbol{z}} P_r\left(\boldsymbol{\psi}|\boldsymbol{z}\right) P_r(\boldsymbol{z}). \tag{4.15}$$

Making the reasonable assumption that $\boldsymbol{y}|\boldsymbol{z}$ and $\boldsymbol{x}|\boldsymbol{z}$ are independent, Equation (4.14) yields:

$$P_r\left(\boldsymbol{z}|\boldsymbol{\psi}\right) \;=\; \frac{P_r\left(\boldsymbol{y}|\boldsymbol{z}\right) P_r\left(\boldsymbol{x}|\boldsymbol{z}\right) P_r(\boldsymbol{z})}{P_r\left(\boldsymbol{\psi}\right)}. \tag{4.16}$$

Dropping $P_r\left(\boldsymbol{\psi}\right)$ as before yields the Form 2 MAP estimate:

$$\hat{\boldsymbol{z}}_2 \;=\; \arg\max_{\boldsymbol{z}} P_r\left(\boldsymbol{y}|\boldsymbol{z}\right) P_r\left(\boldsymbol{x}|\boldsymbol{z}\right) P_r(\boldsymbol{z}). \tag{4.17}$$

Another application of Bayes rule to Equation (4.16) gives the following representation:

$$P_r\left(\boldsymbol{z}|\boldsymbol{\psi}\right) \;=\; \frac{P_r\left(\boldsymbol{y}|\boldsymbol{z}\right)P_r\left(\boldsymbol{z}|\boldsymbol{x}\right)P_r(\boldsymbol{x})}{P_r\left(\boldsymbol{\psi}\right)},$$

from which $P_r\left(\boldsymbol{\psi}\right)$ and $P_r(\boldsymbol{x})$ may be dropped to obtain the Form 1 MAP estimate:

$$\hat{\boldsymbol{z}}_1 \;=\; \arg\max_{\boldsymbol{z}} P_r\left(\boldsymbol{y}|\boldsymbol{z}\right)P_r\left(\boldsymbol{z}|\boldsymbol{x}\right). \tag{4.18}$$

4.5 Derivation of Matrix Equations

From Section 4.4, the Form 1 MAP estimate of $\boldsymbol{z}$ in terms of probability density functions is given by:

$$\hat{\boldsymbol{z}} \;=\; \arg\max_{\boldsymbol{z}} P_r(\boldsymbol{y}|\boldsymbol{z})P_r(\boldsymbol{z}|\boldsymbol{x}), \tag{4.19}$$

A Bayesian general linear model is assumed:

$$\boldsymbol{y} \;=\; W\boldsymbol{z}+\mathbf{n},$$

meaning that the noise vector $\mathbf{n}$ is independent of $\boldsymbol{z}$ and has a Gaussian distribution with mean 0 and covariance C_n. This leads to the following probability density function for $\boldsymbol{y}|\boldsymbol{z}$:

$$P_r(\boldsymbol{y}|\boldsymbol{z}) \;=\; \frac{1}{\sqrt{(2\pi)^{MP}|C_n|}}\exp\left[-\frac{1}{2}(\boldsymbol{y}-W\boldsymbol{z})^T C_n^{-1}(\boldsymbol{y}-W\boldsymbol{z})\right]. \tag{4.20}$$

We also assume the Bayesian general linear model $\boldsymbol{x}=S\boldsymbol{z}+\boldsymbol{\eta}$ which implies that the joint distribution of $(\boldsymbol{x},\boldsymbol{z})$ is multivariate Gaussian. By Theorem 10.2 of [17], this implies that the conditional distribution function of $\boldsymbol{z}|\boldsymbol{x}$ is multivariate Gaussian as well. The relationship between the joint and conditional distributions functions is presented below.

Let $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_z$ denote the means of $\boldsymbol{x}$ and $\boldsymbol{z}$ and let $C_{(x,z)}$ denote the covariance matrix of $(\boldsymbol{x},\boldsymbol{z})$. Then,

$$P_r(\boldsymbol{x},\boldsymbol{z}) \;=\; \frac{1}{\sqrt{(2\pi)^{N+NP}|C_{(x,z)}|}}\exp\left[-\frac{1}{2}\begin{pmatrix}\boldsymbol{x}-\boldsymbol{\mu}_x\\ \boldsymbol{z}-\boldsymbol{\mu}_z\end{pmatrix}^T C_{(x,z)}^{-1}\begin{pmatrix}\boldsymbol{x}-\boldsymbol{\mu}_x\\ \boldsymbol{z}-\boldsymbol{\mu}_z\end{pmatrix}\right],$$

where $|C_{(x,z)}|$ denotes the determinant of $C_{(x,z)}$.

The covariance matrix $C_{(x,z)}$ is an $(N+NP)\times(N+NP)$ real symmetric positive definite matrix. We partition $C_{(x,z)}$ as follows:

$$C_{(x,z)} \;=\; \begin{pmatrix}C_x & C_{zx}^T\\ C_{zx} & C_z\end{pmatrix},$$

where C_x is $N \times N$, C_{zx} is $NP \times N$, and C_z is $NP \times NP$. The sub matrices C_x and C_z are positive definite symmetric since $C_{(x,z)}$ is.

By Theorem 10.2 of [17], the conditional mean vector and covariance matrix are given by:

$$\begin{aligned} \boldsymbol{\mu}_{z|x} &= \boldsymbol{\mu}_z + C_{zx}C_x^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x) && (4.21) \\ C_{z|x} &= C_z - C_{zx}C_x^{-1}C_{zx}^T. && (4.22) \end{aligned}$$

From the model $\boldsymbol{x} = S\boldsymbol{z} + \eta$, we have the following relationships (see [17], p.326):

$$\begin{aligned} \boldsymbol{\mu}_x &= S\boldsymbol{\mu}_z && (4.23) \\ C_x &= SC_zS^T + C_\eta && (4.24) \\ C_{zx} &= C_zS^T, && (4.25) \end{aligned}$$

implying that the conditional mean and covariance are:

$$\begin{aligned} \boldsymbol{\mu}_{z|x} &= \boldsymbol{\mu}_z + C_zS^T\left[SC_zS^T + C_\eta\right]^{-1}(\boldsymbol{x} - S\boldsymbol{\mu}_z) && (4.26) \\ C_{z|x} &= C_z - C_zS^T\left[SC_zS^T + C_\eta\right]^{-1}SC_z. && (4.27) \end{aligned}$$

Therefore, the conditional PDF of $\boldsymbol{z}|\boldsymbol{x}$ is given by:

$$P_r(\boldsymbol{z}|\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^{NP}|C_{z|x}|}}\exp\left[-\frac{1}{2}\left(\boldsymbol{z} - \boldsymbol{\mu}_{z|x}\right)^T C_{z|x}^{-1}\left(\boldsymbol{z} - \boldsymbol{\mu}_{z|x}\right)\right], \quad (4.28)$$

where $\boldsymbol{\mu}_{z|x}$ and $C_{z|x}$ are given by Equations (4.26) and (4.27).

It is seen from Equations (4.20), and (4.28) that maximizing the product $P_r(\boldsymbol{y}|\boldsymbol{z})P_r(\boldsymbol{z}|\boldsymbol{x})$ of Equation (4.19) is equivalent to minimizing the cost function:

$$F_1(\boldsymbol{z}) = \frac{1}{2}(\boldsymbol{y} - W\boldsymbol{z})^T C_n^{-1}(\boldsymbol{y} - W\boldsymbol{z}) + \frac{1}{2}\left(\boldsymbol{z} - \boldsymbol{\mu}_{z|x}\right)^T C_{z|x}^{-1}\left(\boldsymbol{z} - \boldsymbol{\mu}_{z|x}\right), \quad (4.29)$$

so that

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} F_1(\boldsymbol{z}). \quad (4.30)$$

The cost function F_1 may be minimized by the standard multivariate calculus technique of taking setting the gradient of F_1 equal to zero and solving for critical points. Two useful results for this purpose are as follows:

$$\begin{aligned} g(\boldsymbol{z}) &= \tfrac{1}{2}(\boldsymbol{z} - \boldsymbol{\mu})^T A(\boldsymbol{z} - \boldsymbol{\mu}) && \Longrightarrow && \bigtriangledown_z g(\boldsymbol{z}) &= A(\boldsymbol{z} - \boldsymbol{\mu}) \\ g(\boldsymbol{z}) &= \tfrac{1}{2}(\boldsymbol{y} - W\boldsymbol{z})^T A(\boldsymbol{y} - W\boldsymbol{z}) && \Longrightarrow && \bigtriangledown_z g(\boldsymbol{z}) &= -W^T A(\boldsymbol{y} - W\boldsymbol{z}). \end{aligned}$$

Applying these yield the following Form 1 matrix equation whose solution is the value of $\hat{\boldsymbol{z}}$ defined in Equation (4.30):

$$\hat{\boldsymbol{z}} = \left[W^T C_n^{-1} W + C_{z|x}^{-1}\right]^{-1}\left[W^T C_n^{-1}\boldsymbol{y} + C_{z|x}^{-1}\boldsymbol{\mu}_{z|x}\right]. \quad (4.31)$$

The *matrix inversion lemma* (see [17], appendix A1.1.3) is:

$$(A+BCD)^{-1} \;=\; A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}, \tag{4.32}$$

where the indicated inverses are assumed to exist and the matrix sizes are compatible with the indicated products. Applying the matrix inversion lemma yields:

$$\hat{\boldsymbol{z}} \;=\; \left[C_{z|x} - C_{z|x}W^T(WC_{z|x}W^T + C_n)^{-1}WC_{z|x}\right](W^TC_n^{-1}\boldsymbol{y} + C_{z|x}^{-1}\boldsymbol{\mu}_{z|x}).$$

Let $E = WC_{z|x}W^T + C_n$ and simplify the above equation as follows:

$$\begin{aligned}
\hat{\boldsymbol{z}} &= \left[C_{z|x} - C_{z|x}W^TE^{-1}WC_{z|x}\right](W^TC_n^{-1}\boldsymbol{y} + C_{z|x}^{-1}\boldsymbol{\mu}_{z|x}) \\
&= \boldsymbol{\mu}_{z|x} + C_{z|x}W^T\left[C_n^{-1}\boldsymbol{y} - E^{-1}WC_{z|x}W^TC_n^{-1}\boldsymbol{y} - E^{-1}W\boldsymbol{\mu}_{z|x}\right] \\
&= \boldsymbol{\mu}_{z|x} + C_{z|x}W^T\left[C_n^{-1}\boldsymbol{y} - E^{-1}\left(E - C_n\right)C_n^{-1}\boldsymbol{y} - E^{-1}W\boldsymbol{\mu}_{z|x}\right] \\
&= \boldsymbol{\mu}_{z|x} + C_{z|x}W^T\left[C_n^{-1}\boldsymbol{y} - \left(I - E^{-1}C_n\right)C_n^{-1}\boldsymbol{y} - E^{-1}W\boldsymbol{\mu}_{z|x}\right] \\
&= \boldsymbol{\mu}_{z|x} + C_{z|x}W^T\left[C_n^{-1}\boldsymbol{y} - C_n^{-1}\boldsymbol{y} + E^{-1}\boldsymbol{y} - E^{-1}W\boldsymbol{\mu}_{z|x}\right] \\
&= \boldsymbol{\mu}_{z|x} + C_{z|x}W^TE^{-1}(\boldsymbol{y} - W\boldsymbol{\mu}_{z|x}).
\end{aligned}$$

Therefore,

$$\hat{\boldsymbol{z}} = \boldsymbol{\mu}_{z|x} + C_{z|x}W^T[WC_{z|x}W^T + C_n]^{-1}(\boldsymbol{y} - W\boldsymbol{\mu}_{z|x}), \tag{4.33}$$

which is the starting point of Section 4.11.

From Section 4.4, the Form 2 MAP estimator in terms of PDF functions is given by:

$$\hat{\boldsymbol{z}} \;=\; \arg\max_{\boldsymbol{z}} P_r(\boldsymbol{y}|\boldsymbol{z})P_r(\boldsymbol{x}|\boldsymbol{z})P_r(\boldsymbol{z}), \tag{4.34}$$

where

$$P_r(\boldsymbol{x}|\boldsymbol{z}) \;=\; \frac{1}{\sqrt{(2\pi)^N|C_\eta|}}\exp\left[-\frac{1}{2}(\boldsymbol{x}-S\boldsymbol{z})^TC_\eta^{-1}(\boldsymbol{x}-S\boldsymbol{z})\right], \tag{4.35}$$

$$P_r(\boldsymbol{z}) \;=\; \frac{1}{\sqrt{(2\pi)^{NP}|C_z|}}\exp\left[-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu}_z)^TC_z^{-1}(\boldsymbol{z}-\boldsymbol{\mu}_z)\right]. \tag{4.36}$$

Analogous to Form 1, the cost function for Form 2 is obtained from Equations (4.34), (4.20), (4.35), and (4.36):

$$\begin{aligned}
F_2(\boldsymbol{z}) &= \frac{1}{2}(\boldsymbol{y}-W\boldsymbol{z})^TC_n^{-1}(\boldsymbol{y}-W\boldsymbol{z}) + \frac{1}{2}(\boldsymbol{x}-S\boldsymbol{z})^TC_\eta^{-1}(\boldsymbol{x}-S\boldsymbol{z}) \\
&+ \frac{1}{2}\left(\boldsymbol{z}-\boldsymbol{\mu}_z\right)^TC_z^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_z\right).
\end{aligned} \tag{4.37}$$

Minimizing the cost function F_2 of Equation (4.37) yields the Form 2 MAP estimate:

$$\hat{\boldsymbol{z}} \;=\; \left[C_z^{-1} + W^TC_n^{-1}W + S^TC_\eta^{-1}S\right]^{-1}\left[C_z^{-1}\boldsymbol{\mu}_z + W^TC_n^{-1}\boldsymbol{y} + S^TC_\eta^{-1}\boldsymbol{x}\right], \tag{4.38}$$

which is identical to Equation (A.1) and is the starting point of Appendix A. In Appendix A, the matrix inversion lemma is applied twice in succession resulting in a simplified version of the above Form 2 matrix equation.

From Section 4.4, the Form 3 MAP estimator in terms of PDF functions is given by:

$$\hat{\boldsymbol{z}} \;=\; \arg\max_{\boldsymbol{z}} P_r\left(\boldsymbol{\psi}|\boldsymbol{z}\right) P_r(\boldsymbol{z}), \quad \text{where } \boldsymbol{\psi} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix}.$$

Making the associations $\boldsymbol{y} \longleftrightarrow \boldsymbol{\psi}$ and $\boldsymbol{z}|\boldsymbol{x} \longleftrightarrow \boldsymbol{z}$, we see from Equation (4.19) that Form 3 and Form 1 have the same structure. If we further associate:

$$\begin{aligned} W &\longleftrightarrow \begin{pmatrix} W \\ S \end{pmatrix} \\ C_n &\longleftrightarrow \begin{pmatrix} C_n & 0 \\ 0 & C_\eta \end{pmatrix}, \end{aligned}$$

then we may use Equation (4.29) to obtain the Form 3 cost function:

$$\begin{aligned} F_3(\boldsymbol{z}) \;&=\; \frac{1}{2}\left[\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix} - \begin{pmatrix} W \\ S \end{pmatrix}\boldsymbol{z}\right]^T \begin{pmatrix} C_n & 0 \\ 0 & C_\eta \end{pmatrix}^{-1} \left[\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix} - \begin{pmatrix} W \\ S \end{pmatrix}\boldsymbol{z}\right] \\ &+\; \frac{1}{2}\left(\boldsymbol{z}-\boldsymbol{\mu}_z\right)^T C_z^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_z\right), \end{aligned}$$

and use Equation (4.33) to obtain the Form 3 matrix equation:

$$\hat{\boldsymbol{z}} = \boldsymbol{\mu}_z + C_z \begin{pmatrix} W \\ S \end{pmatrix}^T \left[\begin{pmatrix} W \\ S \end{pmatrix} C_z \begin{pmatrix} W \\ S \end{pmatrix}^T + \begin{pmatrix} C_n & 0 \\ 0 & C_\eta \end{pmatrix}\right]^{-1} \left[\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix} - \begin{pmatrix} W \\ S \end{pmatrix}\boldsymbol{\mu}_z\right]. \tag{4.39}$$

We have assumed that $\boldsymbol{\eta}$ and $\mathbf{n}$ are independent in order to use $\begin{pmatrix} C_n & 0 \\ 0 & C_\eta \end{pmatrix}$ as the covariance matrix of $\boldsymbol{\psi}|\boldsymbol{z}$.

Since Form 1 is the only form that is applicable when the model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is *not* assumed, the following theorem proves that it is sufficient to program just Form 1 in order to incorporate all three forms of the MAP algorithm.

Theorem 4.5.1 *Under the assumption $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$, the three forms of MAP estimators defined above are equivalent and each is equal to $E(\boldsymbol{z}|\boldsymbol{\psi})$.*

Proof. The Form 2 and Form 3 MAP estimates are easily seen to be identical by checking that their associated cost functions $F_2(\boldsymbol{z})$ and $F_3(\boldsymbol{z})$ are identical. We may prove that the Form 1 and Form 2 MAP estimates are equivalent either by proving that the gradient of the difference of their cost functions is zero, or by proving the equivalence of Equations (4.31) and (4.38) directly. The latter approach is taken below.

Applying the matrix inversion lemma to Equation (4.27) yields:

$$C_{z|x}^{-1} = C_z^{-1} + S^T C_\eta^{-1} S,$$

and substituting this into Equation (4.31) gives:

$$\hat{\boldsymbol{z}} = \left[W^T C_n^{-1} W + C_z^{-1} + S^T C_\eta^{-1} S\right]^{-1} \left[W^T C_n^{-1} \boldsymbol{y} + \left(C_z^{-1} + S^T C_\eta^{-1} S\right) \boldsymbol{\mu}_{z|x}\right].$$

Comparing the above with Equation (4.38), we see that the Form 1 and Form 2 MAP estimates are equivalent provided that:

$$\left(C_z^{-1} + S^T C_\eta^{-1} S\right) \boldsymbol{\mu}_{z|x} = C_z^{-1} \boldsymbol{\mu}_z + S^T C_\eta^{-1} \boldsymbol{x},$$

where $\boldsymbol{\mu}_{z|x}$ is given by Equation (4.26). The following calculation verifies this:

$$\begin{aligned}
\left(C_z^{-1} + S^T C_\eta^{-1} S\right) \boldsymbol{\mu}_{z|x} &= \left(C_z^{-1} + S^T C_\eta^{-1} S\right) \left[\boldsymbol{\mu}_z + C_z S^T \left[S C_z S^T + C_\eta\right]^{-1} (\boldsymbol{x} - S\boldsymbol{\mu}_z)\right] \\
&= C_z^{-1} \boldsymbol{\mu}_z + S^T \left[S C_z S^T + C_\eta\right]^{-1} (\boldsymbol{x} - S\boldsymbol{\mu}_z) + S^T C_\eta^{-1} S \boldsymbol{\mu}_z \\
&\quad + S^T C_\eta^{-1} S C_z S^T \left[S C_z S^T + C_\eta\right]^{-1} (\boldsymbol{x} - S\boldsymbol{\mu}_z) \\
&= C_z^{-1} \boldsymbol{\mu}_z + S^T \left[S C_z S^T + C_\eta\right]^{-1} (\boldsymbol{x} - S\boldsymbol{\mu}_z) + S^T C_\eta^{-1} S \boldsymbol{\mu}_z \\
&\quad + S^T C_\eta^{-1} \left[\left(S C_z S^T + C_\eta\right) - C_\eta\right] \left[S C_z S^T + C_\eta\right]^{-1} (\boldsymbol{x} - S\boldsymbol{\mu}_z) \\
&= C_z^{-1} \boldsymbol{\mu}_z + S^T \left[S C_z S^T + C_\eta\right]^{-1} (\boldsymbol{x} - S\boldsymbol{\mu}_z) + S^T C_\eta^{-1} S \boldsymbol{\mu}_z \\
&\quad + \left[S^T C_\eta^{-1} - S^T C_\eta^{-1} C_\eta \left[S C_z S^T + C_\eta\right]^{-1}\right] (\boldsymbol{x} - S\boldsymbol{\mu}_z) \\
&= C_z^{-1} \boldsymbol{\mu}_z + S^T C_\eta^{-1} \boldsymbol{x}.
\end{aligned}$$

Since we have shown that all three forms are identical, the fact that each is equal to $E(\boldsymbol{z}|\boldsymbol{\psi})$ is established by showing it for any one of the forms. A straightforward application of Theorem 10.3 of Kay [17] shows that the Form 3 MAP estimate given by Equation (4.39) is equal to $E(\boldsymbol{z}|\boldsymbol{\psi})$. ∎

4.6 Estimation of the Spectral Response Matrix

In Section 4.7, we will use the relationship $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ to estimate the covariance matrix $C_{\tilde{z}|x}$ in terms of the covariance matrix $C_{\tilde{z}}$. For this, it will be important to know how the spectral response matrix S is estimated and what its resulting structure is.

For each low resolution hyper-pixel $\boldsymbol{y}_m$, let $\bar{x}_m$ denote the average of the corresponding broadband image data:

$$\bar{x}_m = \frac{1}{L} \sum_{j=1}^{L} x_{q((m-1)L+j)}. \tag{4.40}$$

Consider the equations:

$$\bar{x}_m = \langle \vec{y}_m, \mathbf{s} \rangle \quad (1 \leq m \leq M), \tag{4.41}$$

where the angle brackets denote the standard inner product and $\mathbf{s} = (s_1, s_2, \ldots, s_P)^T$. In matrix form Equation (4.41) is:

$$Y\mathbf{s} = \bar{\boldsymbol{x}}, \tag{4.42}$$

where $\bar{\boldsymbol{x}} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_M)^T$ and Y is the $M \times P$ matrix determined by placing $\vec{y}_m^T$ in row m for $1 \leq m \leq M$.

Equation (4.42) is solved in the least squares sense, subject to $\sum_{j=1}^{P} s_j = 1$ and $s_j \geq 0$ for $1 \leq j \leq P$, in order to determine a normalized spectral response vector $\mathbf{s}$. The normalized spectral response vector is used to form an $N \times NP$ matrix $\widetilde{S} = SQ^T$ as follows:

$$\widetilde{S} \;=\; \bigoplus_{n=1}^{N} \mathbf{s}^T \;=\; I_N \otimes \mathbf{s}^T. \tag{4.43}$$

Observe that for the no noise case, $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is equivalent to:

$$x_i = \langle \vec{z}_i, \mathbf{s} \rangle \quad (1 \leq i \leq N),$$

which may be viewed as Equation (4.41) extended from low resolution hyper-pixels to high resolution hyper-pixels. Also observe that the assumed form of S dictates that x_i is determined solely from η_i and the hyper pixel $\vec{z}_i$ corresponding to x_i. Thirdly, note that $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ determines $\boldsymbol{x} - \boldsymbol{\eta}$ in super-pixel ordering while $\boldsymbol{x} = \widetilde{S}\boldsymbol{z} + \boldsymbol{\eta}$ determines it in lexicographical order.

In MATLAB, the computation of $\mathbf{s}$ is given by:

$$\mathbf{s} = \texttt{LSQLIN}(Y, \bar{\boldsymbol{x}}, \texttt{[ ]}, \texttt{[ ]}, \texttt{ones}(1, P), 1, \texttt{zeros}(1, P), \texttt{ones}(1, P));$$

where `LSQLIN` is a function available in the MATLAB optimization toolbox.

Since

$$||Y\mathbf{s} - \bar{\boldsymbol{x}}||^2 = \mathbf{s}^T Y^T Y \mathbf{s} - 2\bar{\boldsymbol{x}}^T Y \mathbf{s} + \bar{\boldsymbol{x}}^T \bar{\boldsymbol{x}},$$

the problem may be formulated as a convex quadratic optimization problem. In this formulation, let $C = Y^T Y$, $\mathbf{d} = -Y^T \bar{\boldsymbol{x}}$ and compute $\mathbf{s}$ such that:

$$\frac{1}{2}\mathbf{s}^T C \mathbf{s} + \mathbf{d}^T \mathbf{s}$$

is minimized subject to the linear constraint $\sum_{j=1}^{P} s_j = 1$ and the bound constraints $s_j \geq 0$, $1 \leq j \leq P$. The freely available C program QLD [20] may be used to solve this equivalent problem. We have determined through numerical experimentation that the computations of C and $\mathbf{d}$ incur excessive round off errors if performed in single precision.

4.7 Estimation of Conditional Covariance from Unconditional Covariance

In this section, we assume the linear model $\boldsymbol{x} = S\boldsymbol{x} + \boldsymbol{\eta}$ holds, which implies that the relationship between the covariance matrices $C_{z|x}$ and C_z given by Equation (4.27) holds:

$$C_{z|x} = C_z - C_z S^T (S C_z S^T + C_\eta)^{-1} S C_z. \tag{4.44}$$

Therefore, the relationship between the covariance matrices $C_{\tilde{z}|x}$ and $C_{\tilde{z}}$ is given by:

$$C_{\tilde{z}|x} = C_{\tilde{z}} - C_{\tilde{z}} \widetilde{S}^T (\widetilde{S} C_{\tilde{z}} \widetilde{S}^T + C_\eta)^{-1} \widetilde{S} C_{\tilde{z}}\,, \tag{4.45}$$

where $\widetilde{S} = SQ^T$, $C_{\tilde{z}} = QC_zQ^T$, $C_{\tilde{z}|x} = QC_{z|x}Q^T$, and $\hat{\boldsymbol{z}}$ gives the super-pixel ordering of $\boldsymbol{z}$ as defined in Section 4.2.

An assumption that will facilitate the parallel implementation of the Form 1 MAP estimate is that $C_{\tilde{z}|x}$ be block diagonal with $P \times P$ blocks. Due to the special form of $\widetilde{S}$ given by Equation (4.43), we will see below that $C_{\tilde{z}|x}$ has this form if $C_{\tilde{z}}$ has it. Accordingly, let

$$C_{\tilde{z}} = \bigoplus_{i=1}^{N} B_i, \tag{4.46}$$

where each B_i is a $P \times P$ matrix. From Equation (4.43), the $N \times N$ matrix $\widetilde{S} C_{\tilde{z}} \widetilde{S}^T$ is given by:

$$\widetilde{S} C_{\tilde{z}} \widetilde{S}^T = \bigoplus_{i=1}^{N} \mathbf{s}^T B_i \mathbf{s}.$$

Assuming that $C_\eta = \sigma_\eta^2 I_N$, it follows that

$$(\widetilde{S} C_{\tilde{z}} \widetilde{S}^T + C_\eta)^{-1} = \bigoplus_{i=1}^{N} d_i^{-1},$$

where $d_i = (\mathbf{s}^T B_i \mathbf{s} + \sigma_\eta^2)$. Therefore Equation (4.44) becomes

$$C_{\tilde{z}|x} = C_{\tilde{z}} - C_{\tilde{z}} \widetilde{S}^T D^{-1} \widetilde{S} C_{\tilde{z}},$$

where $D = \bigoplus_{i=1}^{N} d_i$. Since $\widetilde{S}^T D^{-1} \widetilde{S} = \bigoplus_{i=1}^{N} d_i^{-1} \mathbf{s}\mathbf{s}^T$, it follows from Equation (4.46) that:

$$C_{\tilde{z}|x} = \bigoplus_{i=1}^{N} \left[B_i - d_i^{-1} B_i \mathbf{s} \left(B_i \mathbf{s} \right)^T \right], \tag{4.47}$$

and we see that the desired structure for $C_{\tilde{z}|x}$ is inherited from $C_{\tilde{z}}$.

Starting with Equation (4.26), the following expression for $\boldsymbol{\mu}_{\tilde{z}|x}$ is obtained in a similar way:

$$\begin{aligned} \boldsymbol{\mu}_{\tilde{z}|x} &= \left(\mu_{\tilde{z}_1|x}^T, \mu_{\tilde{z}_2|x}^T, \ldots, \mu_{\tilde{z}_N|x}^T\right)^T, \\ \mu_{\tilde{z}_i|x} &= \mu_i + (x_i - \mathbf{s}^T\mu_i) d_i^{-1} B_i \mathbf{s}, \end{aligned} \tag{4.48}$$

where μ_i is a $P \times 1$ vector that denotes the mean of the high resolution hyper-pixel $\tilde{z}_i = \vec{z}_{q(i)}$. Note that the same coefficient $d_i^{-1} B_i \mathbf{s}$ appears in both (4.47) and (4.48) which may be utilized for computational efficiency.

For $1 \leq m \leq M$ and $1 \leq j \leq L$, define the $P \times P$ matrix $B_{m,j} = B_{j+(m-1)L}$. The $B_{m,j}$ merely represent an alternative indexing of the B_i, convenient when identifying super-pixel components. Thus,

$$C_{\tilde{z}} = \bigoplus_{m=1}^{M} \widehat{B}_m, \tag{4.49}$$

$$\widehat{B}_m = \bigoplus_{j=1}^{L} B_{m,j}. \tag{4.50}$$

A similar notation is used for $C_{\tilde{z}|x}$:

$$C_{\tilde{z}|x} = \bigoplus_{m=1}^{M} \widehat{G}_m, \tag{4.51}$$

$$\widehat{G}_m = \bigoplus_{j=1}^{L} G_{m,j}. \tag{4.52}$$

Equation (4.47) implies:

$$G_{m,j} = B_{m,j} - \left(\mathbf{s}^T B_{m,j} \mathbf{s} + \sigma_\eta^2\right)^{-1} \left(B_{m,j}\mathbf{s}\right) \left(B_{m,j}\mathbf{s}\right)^T. \tag{4.53}$$

Starting with estimates $B_{m,j}$ we use Equation (4.53) to obtain $G_{m,j}$ which results in an estimation of $C_{\tilde{z}|x}$. Thus, obtaining a good estimate of $C_{\tilde{z}|x}$ comes down to obtaining good estimates of the $B_{m,j}$. There is a caveat however: the matrix given by Equation (4.53) is singular when $\sigma_\eta = 0$. To see this, let $a_{m,j} = \mathbf{s}^T B_{m,j} \mathbf{s}$ and $\mathbf{v} = B_{m,j}\mathbf{s}$. It is easily verified that $a_{m,j}$ is an eigenvector of $B_{m,j}\mathbf{s}\mathbf{s}^T$ with eigenvector $\mathbf{v}$. Therefore, $\det(a_{m,j} I_P - B_{m,j}\mathbf{s}\mathbf{s}^T) = 0$, which implies that $G_{m,j}$ is singular when $\sigma_\eta = 0$.

The Form 1, 2, and 3 MAP estimates are sensitive to a condition we refer to as the *special case* condition (or assumption). Explicitly, this is the condition:

$$B_{m,j} = B_{m,1} \qquad (1 \leq m \leq M,\ 1 \leq j \leq L).$$

We colloquially refer to this by stating that the $P \times P$ diagonal blocks of $C_{\tilde{z}}$ are identical within super-pixels. It is readily seen from Equation (4.53) that $C_{\tilde{z}|x}$ inherits the special case condition from $C_{\tilde{z}}$.

4.8 Consequences of Making the Conditional Covariance Matrix Diagonal

In this section we discuss how making $C_{\tilde{z}|x}$ a diagonal matrix implies an undesired restriction on the spectral response matrix S. In particular, such a diagonal structure imposes restrictions that are most likely inconsistent with an estimation of S such as that given in Section 4.6.

We begin by examining Equation (4.53) for the case that $B_{m,j}$ is a diagonal matrix. In this case, it is shown below (see Theorem 4.8.1) that $\mathbf{s}$ has only one nonzero entry, which must be 1 due to the normalization $\sum_{j=1}^{P} s_j = 1$. As another case (see Theorem 4.8.2), assuming (4.53) is diagonal when $P = 2$ leads to the result that $s_1 s_2$ is a multiple of σ_η^2 where the multiple depends only on the entries of $B_{m,j}$. Either case leads to the conclusion that $G_{m,j}$ (and hence the conditional covariance matrix $C_{\tilde{z}|x}$) may not be taken as a diagonal matrix without limiting the entries of $\mathbf{s}$ in a manner most likely inconsistent with an estimation of S such as that given in Section 4.6.

Theorem 4.8.1 *If $G_{m,j}$ and $B_{m,j}$ are diagonal then $\mathbf{s} = \mathbf{e}_i$ for some $i \in \{1, 2, \ldots, P\}$ where $\mathbf{e}_i$ denotes the unit vector with a one in its i^{th} entry and zeros elsewhere.*

Proof. From Equation (4.53), we see that $B_{m,j}\mathbf{s}(B_{m,j}\mathbf{s})^T$ is diagonal. Let $\boldsymbol{\upsilon} = B_{m,j}\mathbf{s}$ so that $\upsilon_n \upsilon_k = 0$ for $n \neq k$. Since $\mathbf{s} \neq 0$, $\boldsymbol{\upsilon} \neq 0$ and we may choose i such that $\upsilon_i \neq 0$. Therefore, $\boldsymbol{\upsilon} = \upsilon_i \mathbf{e}_i$. However, $\boldsymbol{\upsilon} = (b_1 s_1, b_2 s_2, \ldots, b_P s_P)^T$ where $B_{m,j} = \text{ diag}(b_1, b_2, \ldots, b_P)$, so it follows that $b_k s_k = 0$ whenever $k \neq i$. Since $B_{m,j}$ is positive definite, $b_k > 0$ for all k. Therefore, $s_k = 0$ for $k \neq i$ and since $\sum_{j=1}^{P} s_j = 1$, $s_i = 1$. ∎

Theorem 4.8.2 *If $G_{m,j}$ is diagonal and $P = 2$, then*

$$s_1 s_2 \;=\; \frac{b_{1,2}\sigma_\eta^2}{|B|}$$

where

$$B = B_{m,j} \;=\; \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{1,2} & b_{2,2} \end{pmatrix}.$$

Proof. Using Equation (4.53), we set the $(1,2)$ entry of $G_{m,j}$ equal to zero:

$$b_{1,2} - (b_{1,1}s_1^2 + 2b_{1,2}s_1 s_2 + b_{2,2}s_2^2 + \sigma_\eta^2)^{-1}(b_{1,1}s_1 + b_{1,2}s_2)(b_{1,2}s_1 + b_{2,2}s_2) \;=\; 0.$$

and the result easily follows. ∎

For the general case when $B_{m,j}$ is any positive definite symmetric matrix and P is any positive integer, we argue heuristically as follows. Setting the off diagonal entries of the

symmetric matrix $G_{m,j}$ equal to zero leads to the following system of $P(P-1)/2$ nonlinear equations in P unknowns:

$$\sum_{\ell=1}^{P}\sum_{n=1}^{P}(b_{i,n}b_{k,\ell}-b_{\ell,n}b_{i,k})s_\ell s_n \;=\; \sigma_\eta^2 b_{i,k}, \quad 1\le i<k\le P.$$

Because of the condition $\sum_{j=1}^{P} s_j = 1$, each of these equations represents a $P-2$ dimensional surface that lies in a $P-1$ dimensional space. The intersection of all $P(P-1)/2$ surfaces represents the set of solutions $\mathbf{s}$ for which $G_{m,j}$ is diagonal. Intersecting these surfaces will result in a solution set of dimension $P-2$ or less, most likely of dimension much less than $P-2$ for sufficiently large P. (We see from the examples given above that solution spaces of dimension 0 occur when $B_{m,j}$ is diagonal or when $P=2$.) We conclude that since the solution set is of dimension $P-2$ or less, a procedure for determining $\mathbf{s}$ such as that of Section 4.6 will most likely be inconsistent with the assumption that conditional covariance matrix $C_{\tilde{z}|x}$ is diagonal.

4.9 Conditional Independence

This section defines two levels of conditional independence and relates them to the assumed structures of associated matrices. The first level factors the probability distribution function of $\boldsymbol{z}|\boldsymbol{x}$ into the conditional distribution functions of $\vec{z}_n|x_n$, where $\vec{z}_n$ is the n^{th} hyper-pixel of $\boldsymbol{z}$, and the second level factors it further into the conditional distribution functions of $z_{p,n}|x_n$ where $\vec{z}_n = (z_{1,n}, z_{2,n}, \ldots, z_{P,n})^T$. The first level of conditional independence follows from assumptions previously made concerning the $P\times P$ diagonal block structure of the covariance matrix $C_{\tilde{z}}$ and the Bayesian general linear model $\boldsymbol{x} = S\boldsymbol{z}+\boldsymbol{\eta}$, while the second level is equivalent to the assumption that the conditional covariance matrix $C_{\tilde{z}|x}$ is diagonal.

Explicitly, the two levels of conditional independence are:

$$\text{Level 1}: \quad P_r(\boldsymbol{z}|\boldsymbol{x}) \;=\; \prod_{n=1}^{N} P_r(\vec{z}_n|x_n), \tag{4.54}$$

$$\text{Level 2}: \; P_r(\vec{z}_n|x_n) \;=\; \prod_{p=1}^{P} P_r(z_{p,n}|x_n), \quad n=1,2,\ldots,N. \tag{4.55}$$

We show below that conditional independence level 1 is implied by the Bayesian general linear model $\boldsymbol{x} = S\boldsymbol{z}+\boldsymbol{\eta}$ and the assumed structures of $C_{\tilde{z}}$ and S. Since $C_{\tilde{z}}$ is block diagonal with $P\times P$ blocks, and the entries of $C_{\tilde{z}}$ corresponding to $\vec{z}_n$ and $\vec{z}_k$ are proportional to the correlations $\rho(z_{i,n}, z_{j,k})$ where $1\le i,j\le P$ and $1\le n,k\le N$, it follows that these correlations are zero whenever $n\ne k$. We express this succinctly as: $\rho(\vec{z}_n, \vec{z}_k) = 0$ whenever $(n\ne k)$.

From Equation (4.24) and $C_z = Q^T C_{\tilde{z}} Q$, it easily follows that:

$$C_x \;=\; \widetilde{S} C_{\tilde{z}} \widetilde{S}^T + C_\eta,$$

where $\widetilde{S} = SQ^T$ is given by Equation (4.43). This and Equation (4.46) imply that C_x is a diagonal matrix:

$$C_x = \bigoplus_{n=1}^{N} \left(\mathbf{s}^T B_n \mathbf{s}\right) + C_\eta,$$

so that $\rho(x_n, x_k) = 0$ whenever $n \neq k$. Similarly, equation (4.25) easily implies:

$$C_{zx} \;=\; Q^T \left(\bigoplus_{n=1}^{N} B_n \mathbf{s} \right),$$

so that $\rho(x_n, \vec{z}_k) = 0$ whenever $n \neq k$. Collecting the above results we have:

$$\rho(x_n, x_k) \;=\; \rho(x_n, \vec{z}_k) \;=\; \rho(\vec{z}_n, \vec{z}_k) \;=\; 0 \qquad (n \neq k).$$

For Gaussian distributions, a zero correlation between two random variables implies independence between them. Therefore,

$$\begin{aligned} P_r(\boldsymbol{x}) &= \prod_{n=1}^{N} P_r(x_n) \\ P_r(\boldsymbol{x}, \boldsymbol{z}) &= \prod_{n=1}^{N} P_r(x_n, \vec{z}_n), \end{aligned}$$

and equation (4.54) readily follows:

$$\begin{aligned} P_r(\boldsymbol{z}|\boldsymbol{x}) &= \frac{P_r(\boldsymbol{x}, \boldsymbol{z})}{P_r(\boldsymbol{x})} \\ &= \prod_{n=1}^{N} \frac{P_r(x_n, \vec{z}_n)}{P_r(x_n)} \\ &= \prod_{n=1}^{N} P_r(\vec{z}_n | x_n). \end{aligned}$$

Since $\vec{z}_1$ and x_1 are independent of $(x_2, x_3, \ldots, x_N)$ we have:

$$\begin{aligned} P_r(\vec{z}_1|\boldsymbol{x}) &= \frac{P_r(\vec{z}_1, \boldsymbol{x})}{P_r(\boldsymbol{x})}, \\ &= \frac{P_r(\vec{z}_1, x_1) P_r(x_2, x_3, \ldots, x_N)}{P_r(x_1) P_r(x_2, x_3, \ldots, x_N)}, \\ &= P_r(\vec{z}_1|x_1), \end{aligned}$$

and similarly,

$$P_r(\vec{z}_n|\boldsymbol{x}) \;=\; P_r(\vec{z}_n|x_n), \qquad (1 \le n \le N).$$

Arguing in the same manner yields:

$$P_r(z_{p,n}|\boldsymbol{x}) \;=\; P_r(z_{p,n}|x_n), \quad (1 \le n \le N,\ 1 \le p \le P).$$

Using the fact that for Gaussian distributions, zero correlation and independence are equivalent, it follows that $C_{\tilde{z}|x}$ is diagonal if and only if

$$P_r(\vec{z}_n|\boldsymbol{x}) \;=\; \prod_{p=1}^{P} P_r(z_{p,n}|\boldsymbol{x}), \quad n = 1, 2, \ldots, N.$$

From the above results it follows that $C_{\tilde{z}|x}$ is diagonal if and only if conditional independence level 2 (i.e. Equation (4.55)) holds for $n = 1, 2, \ldots, N$.

As remarked in Section 4.8, the assumption that $C_{\tilde{z}|x}$ is diagonal is likely to be inconsistent with the estimation given in Section 4.6, making level 2 conditional independence an undesirable assumption.

4.10 Principal Component Analysis

Principal component analysis (PCA) starts with the construction of a $P \times P$ "spectral" covariance matrix C that represents the covariance among the entries of a high resolution hyper-pixel. Since $\boldsymbol{z}$ is presumed unknown, the hyper-pixels of $\boldsymbol{y}$ may be used to compute an estimate of C. Using an eigenvalue decomposition algorithm, an orthonormal set of eigenvectors of C is obtained and is used to construct an orthogonal matrix E whose columns are the eigenvectors. The unknown high resolution hyper-pixels $\boldsymbol{z}_i$ $(i = 1, 2, \ldots, N)$ are related to a different set of unknowns $\bar{\boldsymbol{z}}_i$ through the transformation $\boldsymbol{z}_i = E\bar{\boldsymbol{z}}_i$. The problem of computing $\boldsymbol{z}$ such that $\boldsymbol{y} = W\boldsymbol{z} + \mathbf{n}$ and $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is transformed into computing $\bar{\boldsymbol{z}}$ such that $\bar{\boldsymbol{y}} = \overline{W}\bar{\boldsymbol{z}} + \bar{\mathbf{n}}$ and $\boldsymbol{x} = \bar{S}\bar{\boldsymbol{z}} + \boldsymbol{\eta}$, where the quantities with bars are defined below.

Let $E_N = \bigoplus_{n=1}^{N} E$ and $E_M = \bigoplus_{m=1}^{M} E$ and define $\bar{\boldsymbol{z}} = E_N^T\boldsymbol{z}$, $\bar{\boldsymbol{y}} = E_M^T\boldsymbol{y}$, $\bar{\mathbf{n}} = E_M^T\mathbf{n}$, $\bar{S} = SE_N$, and $\overline{W} = E_M^T W E_N$. Since $E^{-1} = E^T$, the same is true of E_N and E_M, making it straightforward to show the equivalence of $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ and $\boldsymbol{x} = \bar{S}\bar{\boldsymbol{z}} + \boldsymbol{\eta}$ as well as the equivalence of $\boldsymbol{y} = W\boldsymbol{z} + \mathbf{n}$ and $\bar{\boldsymbol{y}} = \overline{W}\bar{\boldsymbol{z}} + \bar{\mathbf{n}}$. Since $\mathbf{n}$ represents white Gaussian noise, so does $\bar{\mathbf{n}}$ and it follows that an image enhancement algorithm for computing $\boldsymbol{z}$ may be applied to obtain $\bar{\boldsymbol{z}}$, where the relevant input quantities are substituted with their barred counterparts. We also observe that since E is orthogonal, $||\bar{\mathbf{n}}|| = ||\mathbf{n}||$ so that noise is not magnified when working in the principal component domain. Although $\bar{\boldsymbol{z}}$ may be used directly for producing images, $\boldsymbol{z}$ may be obtained from $\boldsymbol{z} = E_N\bar{\boldsymbol{z}}$ if desired.

The spectral response matrices in their respective domains being related by $\bar{S} = SE_N$ implies that the $P \times 1$ corresponding spectral response vectors are related by:

$$\bar{\mathbf{s}} = E^T \mathbf{s} \tag{4.56}$$

If the PSF matrix W satisfies assumption 2 of Section 4.3 (i.e. does not vary spectrally), then $\overline{W} = W$. This is important since many of the image enhancement algorithms we have examined take advantage of special assumed forms of W for efficiency purposes. In particular, the current version of the Form 1 MAP algorithm makes use of PSF assumptions 1, 2, and 3 of Section 4.3 for efficient coding. We prove below that assumption 2 implies $\overline{W} = W$.

As in Section 4.3, we partition W into $P \times P$ blocks $W_{i,j}$ where $1 \leq i \leq M$ and $1 \leq j \leq N$. It is easily seen that the corresponding (i, j) block of $\overline{W}$ is given by $E^T W_{i,j} E$. The assumption that W does not vary spectrally is equivalent to $W_{i,j}$ being a multiple of I_P for all (i, j). Therefore, $W_{i,j}$ commutes with E. Since E is orthogonal, $E^T E = I_P$. Hence, the (i, j) block of $\overline{W}$ is $W_{i,j}$ implying that $\overline{W} = W$.

Since the eigenvectors of E corresponding to the smaller eigenvalues are not expected to contribute to the enhancement of images, we now examine a "limited" PCA where only the largest eigenvalues are computed. The intent is to improve performance without degrading image quality. Accordingly, choose p $(1 \leq p \leq P)$ and compute the eigenvectors of C belonging to the largest p eigenvectors. Let E be the $P \times p$ matrix defined by $E = (E_1, E_2, \ldots, E_p)$ where the E_i are $P \times 1$ eigenvectors sorted by eigenvalue from largest to smallest. Using the same definitions given in above, E_N becomes $NP \times Np$, E_M is $MP \times Mp$, $\bar{\boldsymbol{z}}$ is $Np \times 1$, $\bar{S}$ is $N \times Np$, $\bar{\boldsymbol{y}}$ is $Mp \times 1$, and $\bar{\mathbf{n}}$ is $Mp \times 1$.

After applying an estimate such as the Form 1 MAP estimate, we obtain an estimate of $\bar{\boldsymbol{z}}$. As before, $\bar{\boldsymbol{z}}$ may be used directly to obtain images. Optionally, the corresponding estimate of $\boldsymbol{z}$ is given by $\hat{\boldsymbol{z}} = E_N \bar{\boldsymbol{z}}$. We note that the resulting size of $\boldsymbol{z}$ is $NP \times 1$ even if $p < P$. For efficiency, the product $E_N \bar{\boldsymbol{z}}$ may be obtained from the product $E[\bar{\boldsymbol{z}}_1, \bar{\boldsymbol{z}}_2, \ldots, \bar{\boldsymbol{z}}_N]$, where each $\bar{\boldsymbol{z}}_i$ is $p \times 1$ and $\bar{\boldsymbol{z}} = (\bar{\boldsymbol{z}}_1; \bar{\boldsymbol{z}}_2; \ldots; \bar{\boldsymbol{z}}_N)$. The resulting $P \times N$ matrix may then be reshaped to a $NP \times 1$ vector to obtain $\hat{\boldsymbol{z}}$.

There is a less efficient, but perhaps more intuitive way to perform limited PCA analysis. In this formulation, $\bar{\boldsymbol{y}}$ is kept as $MP \times 1$, but with each low resolution hyper-pixel having a zero in its last $P - p$ entries. We also leave E as a $P \times P$ matrix but take its last $P - p$ columns to be zero vectors of length P. This results in essentially the same $\hat{\boldsymbol{z}}$ as above, but now each $\bar{\boldsymbol{z}}_i$ is a $P \times 1$ vector with zeros in the last $P - p$ entries. The resulting limited PCA computation of $\hat{\boldsymbol{z}}$ is identical to the one obtained above. The advantages of the former method is not only that it takes less storage, but that it also takes less time to compute, because calculations that multiply zero times itself are eliminated.

The advantage of PCA space processing lies in the fact that most of the signal energy lies in the PCA subspace corresponding to the top eigenvalues. The lower principal component dimensions can be treated with a simpler algorithm, such as interpolation, or even set to zero

if they are deemed to be primarily noise. Processing in a lower dimension PCA subspace reduces the size of the matrix inverse required in the closed form solution and requires the estimation of fewer statistical parameters. In addition to the benefit in terms of processing time, this tends to lead to covariance estimates that are better conditioned, making the resulting MAP algorithm more robust.

4.11 Analysis of the Form 1 MAP Estimate

Starting with the Form 1 MAP estimate presented in Section 4.5, this section derives an equivalent expression that is suitable for parallel implementation. Assuming the special case where the $P \times P$ blocks of $C_{\tilde{z}}$ are identical within super-pixels, we then prove that the matrix appearing as an inverse in the new expression is singular if and only if $\sigma_n = \sigma_\eta = 0$ (i.e. there is no noise). A separate expression is developed under the noise free and special case assumptions and in this expression, no matrix inverses whatsoever appear. *Having no matrix inverses to compute leads to an algorithm whose complexity is low enough to be considered for real time implementation on a parallel computer system.*

In order to facilitate parallel computations, each of these Form 1 MAP expressions are broken down into super-pixel components. This allows the work to be distributed among the available processors in a convenient and efficient fashion. Throughout this section, the three conditions of Section 4.3 are assumed and the definitions of $B_{m,j}$ given by Equations (4.49) and (4.50) are adopted.

As derived in Section 4.5, the Form 1 MAP estimator is given by:

$$\hat{\boldsymbol{z}} = \boldsymbol{\mu}_{z|x} + C_{z|x}W^T[WC_{z|x}W^T + C_n]^{-1}(\boldsymbol{y} - W\boldsymbol{\mu}_{z|x}). \tag{4.57}$$

If

$$\tilde{\boldsymbol{z}} = Q\boldsymbol{z}, \quad \boldsymbol{\mu}_{\tilde{z}|x} = Q\boldsymbol{\mu}_{z|x}, \quad \text{and} \quad C_{\tilde{z}|x} = QC_{z|x}Q^T,$$

where Q is the super-pixel permutation matrix defined in Section 4.2, then Equation (4.57) becomes

$$\tilde{\boldsymbol{z}} = \boldsymbol{\mu}_{\tilde{z}|x} + C_{\tilde{z}|x}\widetilde{W}^T[\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n]^{-1}(\boldsymbol{y} - \widetilde{W}\boldsymbol{\mu}_{\tilde{z}|x}), \tag{4.58}$$

and we see that Equations (4.57) and (4.58) have the same form. Equation (4.58) is much easier work with than (4.57) due to the simple structure of $\widetilde{W}$. The estimate $\hat{\boldsymbol{z}}$ is given by $\hat{\boldsymbol{z}} = Q^T\tilde{\boldsymbol{z}}$ which may be implemented in an efficient manner by permuting the hyper-pixels of $\tilde{\boldsymbol{z}}$ as follows:

$$\hat{\boldsymbol{z}} = (\tilde{z}_{q^{-1}(1)}; \tilde{z}_{q^{-1}(2)}; \ldots; \tilde{z}_{q^{-1}(N)}),$$

where the semi-colon denotes vertical concatenation and q denotes the super-pixel permutation corresponding to Q defined in Section 4.2. In the notation above, $\tilde{z}_{q^{-1}(n)} = \vec{z}_n$ represents the n^{th} hyper-pixel for $n = 1, 2, \ldots N$.

A simplifying assumption, which appears to be necessary for efficient parallel computations, is that $C_{\tilde{z}} = QC_zQ^T$ be block diagonal with $P \times P$ blocks. From Section 4.7, we know that $C_{\tilde{z}}$ having this structure implies that $C_{\tilde{z}|x} = QC_{z|x}Q^T$ will as well. Since pre multiplication by Q followed by post multiplication by Q^T permutes the $P \times P$ blocks along the macro diagonal of such matrices, it follows that C_z, $C_{\tilde{z}|x}$, and $C_{z|x}$ all inherit the same $P \times P$ block diagonal structure from $C_{\tilde{z}}$.

Let $L = N/M$ and partition $C_{\tilde{z}|x}$ into diagonal blocks $\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_M$ where each $\widehat{G}_m$ is a $LP \times LP$ block diagonal matrix with $P \times P$ blocks. Thus,

$$C_{\tilde{z}|x} = \bigoplus_{m=1}^{M} \widehat{G}_m, \tag{4.59}$$

and each $\widehat{G}_m$ for $m = 1, 2, \ldots, M$ has the form:

$$\widehat{G}_m = \bigoplus_{j=1}^{L} G_{m,j}, \tag{4.60}$$

where $G_{m,j}$ is a $P \times P$ matrix. It follows from Equations (4.12) and (4.59) that

$$\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T = \bigoplus_{m=1}^{M} \overline{G}_m$$

where $\overline{G}_m$ is the $P \times P$ matrix given by:

$$\overline{G}_m \;\; = \;\; (\vec{\omega}_m^T \otimes I_P)\widehat{G}_m(\vec{\omega}_m \otimes I_P) \;\; = \;\; \textstyle\sum_{j=1}^{L} \hat{\omega}_{m,j}^2 G_{m,j}. \tag{4.61}$$

Due to the simplifying assumption that the noise covariance matrix C_n is a multiple σ_n^2 of the $MP \times MP$ identity matrix, the inverted matrix of Equation (4.58) becomes:

$$[\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n]^{-1} = \bigoplus_{m=1}^{M} (\overline{G}_m + \sigma_n^2 I_P)^{-1}. \tag{4.62}$$

Similarly, the $NP \times MP$ product $C_{\tilde{z}|x}\widetilde{W}^T$ appearing in Equation (4.58) is given by:

$$C_{\tilde{z}|x}\widetilde{W}^T = \bigoplus_{m=1}^{M} \widehat{G}_m(\vec{\omega}_m \otimes I_P). \tag{4.63}$$

The matrix $C_{\tilde{z}|x}\widetilde{W}^T$ of Equation (4.63) is a non square $(NP \times MP)$ matrix that consists of M blocks on the diagonal, each of size $LP \times P$.

Using the superscript (1) to designate a quantity that pertains to the Form 1 MAP estimate, define the $NP \times MP$ matrix $A^{(1)}$ by:

$$A^{(1)} \;\; = \;\; C_{\tilde{z}|x}\widetilde{W}^T[\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n]^{-1},$$

so that the Form 1 MAP estimate is given by:

$$\tilde{\boldsymbol{z}} \;=\; \boldsymbol{\mu}_{\tilde{z}|x} + A^{(1)}(\boldsymbol{y} - \widetilde{W}\boldsymbol{\mu}_{\tilde{z}|x}).$$

Putting Equations (4.62) and (4.63) together yield:

$$\begin{aligned} A^{(1)} &= \bigoplus_{m=1}^{M} \widehat{G}_m(\vec{\omega}_m \otimes I_P)(\bar{G}_m + \sigma_n^2 I_P)^{-1} \\ &= \bigoplus_{m=1}^{M} A_m^{(1)}. \end{aligned}$$

The $LP \times P$ matrix $A_m^{(1)}$ is defined by:

$$A_m^{(1)} \;=\; \widehat{G}_m(\vec{\omega}_m \otimes I_P)(\bar{G}_m + \sigma_n^2 I_P)^{-1}, \tag{4.64}$$

which when written out becomes:

$$A_m^{(1)} \;=\; \begin{bmatrix} \hat{\omega}_{m,1} G_{m,1}(\bar{G}_m + \sigma_n^2 I_P)^{-1} \\ \hat{\omega}_{m,2} G_{m,2}(\bar{G}_m + \sigma_n^2 I_P)^{-1} \\ \vdots \\ \hat{\omega}_{m,L} G_{m,L}(\bar{G}_m + \sigma_n^2 I_P)^{-1} \end{bmatrix}. \tag{4.65}$$

From Equation (4.58), we see that the Form 1 estimator is given by:

$$\tilde{\boldsymbol{z}} \;=\; \boldsymbol{\mu}_{\tilde{z}|x} + \left(\bigoplus_{m=1}^{M} A_m^{(1)} \right) (\boldsymbol{y} - \widetilde{W}\boldsymbol{\mu}_{\tilde{z}|x}).$$

Breaking this up into M super-pixel components yields:

$$\tilde{\boldsymbol{z}}_m \;=\; \boldsymbol{\mu}_{\tilde{z}_m|x} + A_m^{(1)} \left[\vec{y}_m - \left(\vec{\omega}_m^T \otimes I_P \right) \boldsymbol{\mu}_{\tilde{z}_m|x} \right], \tag{4.66}$$

or equivalently:

$$\tilde{\boldsymbol{z}}_m \;=\; \left[I_{LP} - A_m^{(1)} \left(\vec{\omega}_m^T \otimes I_P \right) \right] \boldsymbol{\mu}_{\tilde{z}_m|x} + A_m^{(1)} \vec{y}_m, \tag{4.67}$$

where $\tilde{\boldsymbol{z}}_m$ and $\boldsymbol{\mu}_{\tilde{z}_m|x}$ are $LP \times 1$ and $\vec{y}_m$ is a $P \times 1$ hyper-pixel. The breakdown of $\tilde{\boldsymbol{z}}$ into M super-pixel vectors is:

$$\tilde{\boldsymbol{z}} \;=\; \left(\tilde{\boldsymbol{z}}_1^T, \tilde{\boldsymbol{z}}_2^T, \ldots, \tilde{\boldsymbol{z}}_M^T \right)^T,$$

and the breakdown of the super-pixel $\tilde{\boldsymbol{z}}_m$ into L hyper-pixels is:

$$\begin{aligned} \tilde{\boldsymbol{z}}_m &= \left(\tilde{z}_{(m-1)L+1}^T, \tilde{z}_{(m-1)L+2}^T, \ldots, \tilde{z}_{mL}^T \right)^T, \\ &= \left(\vec{z}_{q((m-1)L+1)}^T, \vec{z}_{q((m-1)L+2)}^T, \ldots, \vec{z}_{q(mL)}^T \right)^T. \end{aligned}$$

Similarly, the breakdown of the conditional mean $\boldsymbol{\mu}_{\tilde{z}|x}$ into M super-pixel components is:

$$\boldsymbol{\mu}_{\tilde{z}|x} \;=\; \left(\boldsymbol{\mu}_{\tilde{z}_1|x}^T, \boldsymbol{\mu}_{\tilde{z}_2|x}^T, \ldots, \boldsymbol{\mu}_{\tilde{z}_M|x}^T\right)^T,$$

where the m^{th} conditional mean super-pixel (derived from Equation (4.26)) is given by:

$$\boldsymbol{\mu}_{\tilde{z}_m|x} \;=\; \boldsymbol{\mu}_{\tilde{z}_m} + \hat{B}_m \widetilde{S}_L^T \left[\widetilde{S}_L \hat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L\right]^{-1} (\boldsymbol{x}_m - \widetilde{S}_L \boldsymbol{\mu}_{\tilde{z}_m}), \tag{4.68}$$

which may be broken down further to hyper-pixels resulting in Equation (4.48). The quantities appearing in Equation (4.68) are defined by:

$$\begin{aligned}
\widetilde{S}_L &= I_L \otimes \mathbf{s}^T, && \left(\widetilde{S}_L \text{ is } L \times LP,\ \widetilde{S} = SQ^T = I_M \otimes \widetilde{S}_L\right), \\
\boldsymbol{x} &= \left(\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_M^T\right)^T, && \left(\boldsymbol{x}_m \text{ is } L \times 1\right), \\
\boldsymbol{\mu}_{\tilde{z}} &= \left(\boldsymbol{\mu}_{\tilde{z}_1}^T, \boldsymbol{\mu}_{\tilde{z}_2}^T, \ldots, \boldsymbol{\mu}_{\tilde{z}_M}^T\right)^T, && \left(\boldsymbol{\mu}_{\tilde{z}_m} \text{ is } LP \times 1\right), \\
C_{\tilde{z}} &= \bigoplus_{m=1}^{M} \hat{B}_m, && \left(\hat{B}_m = \bigoplus_{j=1}^{L} B_{m,j} \text{ is } LP \times LP\right).
\end{aligned}$$

The inverted quantity in Equation (4.68) reduces to:

$$\widetilde{S}_L \hat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L \;=\; \bigoplus_{j=1}^{L} \left(\mathbf{s}^T B_{m,j} \mathbf{s} + \sigma_\eta^2\right).$$

Therefore, the computation of $\boldsymbol{\mu}_{\tilde{z}_m|x}$ reduces to inverting diagonal matrices.

The computation of $\tilde{\boldsymbol{z}}_m$ using Equations (4.64), (4.67), and (4.68) may be accomplished in parallel by assigning a different set of m to each processing unit. This means that no parallel matrix algorithms need be considered, making coding a task considerably simpler than it would otherwise be. It is important to remark that Equations (4.64) and (4.67) apply even if the relationship $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is *not* assumed. This facilitates the construction of a single computer code that applies to the case where $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is assumed as well as to the case where it is not.

We now assume the special case where the $P \times P$ diagonal blocks of $C_{\tilde{z}}$ (and hence $C_{\tilde{z}|x}$) are identical within super-pixels. Thus, for each $m = 1, 2, \ldots, M$:

$$\begin{array}{ccccccc}
B_{m,1} & = & B_{m,2} & = & \cdots & = & B_{m,L}, \\
G_{m,1} & = & G_{m,2} & = & \cdots & = & G_{m,L}.
\end{array}$$

Under the special case condition, we may characterize the definiteness of the inverted matrix of Equation (4.58) in terms of σ_n and σ_η. For this purpose, we establish the following lemma.

Lemma 4.11.1 *Given any nonzero vector v of length P, the matrix $\Gamma = I_P - \dfrac{vv^T}{v^T v + \epsilon}$ is positive definite if $\epsilon > 0$, and nonnegative definite if $\epsilon = 0$.*

Proof. First assume that $\epsilon > 0$ and let $\Psi = \dfrac{vv^T}{v^Tv + \epsilon}$ so that

$$\|\Psi\|_F \;=\; \left\| \frac{vv^T}{v^Tv+\epsilon} \right\|_F \;=\; \frac{\|vv^T\|_F}{v^Tv+\epsilon} \;=\; \frac{v^Tv}{v^Tv+\epsilon} \;< 1.$$

where $\|\|_F$ denotes the Frobenius norm. It follows that $\Gamma = I_P - \Psi$ is nonsingular and the inverse of Γ is given by the Neumann series:

$$\Gamma^{-1} \;=\; \sum_{n=0}^{\infty} \Psi^n.$$

For any nonzero vector x of length P,

$$x^T\Psi x \;=\; \frac{x^Tvv^Tx}{v^Tv+\epsilon} \;=\; \frac{\|x^Tv\|_2}{v^Tv+\epsilon} \;>\; 0,$$

which proves Ψ is positive definite. Since Ψ is symmetric, it follows that Ψ^n is positive definite for all nonnegative integers n, which proves from the Neumann series that Γ^{-1} and hence Γ is positive definite.

Since a matrix is positive definite if and only if its eigenvalues are positive, and the eigenvalues of a matrix are continuous functions of its entries, it follows that the eigenvalues of Γ, are positive or zero when $\epsilon = 0$. This proves that Γ is nonnegative definite if $\epsilon = 0$.

The fact that a zero eigenvalue of Γ actually occurs when $\epsilon = 0$ is established by observing that v^Tv is an eigenvalue of vv^T with eigenvector v, making $\det(v^TvI_P - vv^T) = 0$. Therefore Γ is singular and hence has at least one zero eigenvalue when $\epsilon = 0$. ∎

Theorem 4.11.2 *Under the special case condition, the (inverted) matrix* $\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n$ *of Equation* (4.58) *is positive definite if either* $\sigma_n > 0$ *or* $\sigma_\eta > 0$ *and is singular if* $\sigma_n = \sigma_\eta = 0$.

Proof. From Equation (4.62), it is equivalent to prove that $\bar{G}_m + \sigma_n^2 I_P$ is singular if $\sigma_n = \sigma_\eta = 0$ and positive definite otherwise. Under the special case assumption, Equation (4.61) becomes:

$$\bar{G}_m \;=\; \gamma_m G_{m,1} \quad \text{where} \quad \gamma_m = \vec{\omega}_m^T\vec{\omega}_m.$$

In Section 4.7, we proved that $G_{m,1}$ is singular when $\sigma_\eta = 0$. It follows that $\bar{G}_m + \sigma_n^2 I_P$ is singular when $\sigma_n = \sigma_\eta = 0$. Therefore, we turn to the case where either $\sigma_n > 0$ or $\sigma_\eta > 0$.

From Equation (4.53),

$$\begin{aligned}
\bar{G}_m + \sigma_n^2 I_P &= \gamma_m G_{m,1} + \sigma_n^2 I_P \\
&= \gamma_m \left[B_{m,1} - \frac{B_{m,1}\mathbf{s}\mathbf{s}^T B_{m,1}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2} \right] + \sigma_n^2 I_P, \\
&= B_{m,1}^{1/2} \left[\gamma_m \left(I_P - \frac{vv^T}{v^Tv + \sigma_\eta^2} \right) + \sigma_n^2 B_{m,1}^{-1} \right] B_{m,1}^{1/2},
\end{aligned}$$

where $v = B_{m,1}^{1/2}\mathbf{s}$. Since the sum of a positive definite matrix with a nonnegative definite matrix is positive definite, Lemma 4.11.1 implies that the matrix quantity within the square brackets of the last expression is positive definite when either $\sigma_n > 0$ or $\sigma_\eta > 0$. From this, it easily follows that $\bar{G}_m + \sigma_n^2 I_P$ is positive definite and the theorem is proven. ∎

Experience with a computer implementation of the Form 1 MAP algorithm shows that the matrix $\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n$ (regardless of whether $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is assumed and regardless of the special case condition) may be poorly conditioned when $\sigma_n = 0$. In fact, taking $\sigma_n > 0$ is a mechanism one can use for diagonally loading $\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n$. When diagonal loading is used in this manner, better performance of the algorithm is observed, as measured by signal to noise ratio determined by comparing the true $\boldsymbol{z}$ (assuming it is known) with the Form 1 MAP estimate $\hat{\boldsymbol{z}}$.

Although $\widetilde{W}C_{\tilde{z}|x}\widetilde{W}^T + C_n$ is singular under the special case and noise free assumptions, that singularity cancels when combined with $C_{\tilde{z}|x}\widetilde{W}^T$ via Equation (4.57). The remainder of this section derives an expression for the Form 1 MAP estimator under the special case and noise free assumptions. We will see that no matrix inversions are required.

Under the noise free assumption ($\sigma_n = \sigma_\eta = 0$), we see from Equation (4.65) that

$$A_m^{(1)} \;=\; \frac{1}{\gamma_m}\left(\vec{\omega}_m \otimes I_P\right), \tag{4.69}$$

and Equation (4.67) becomes:

$$\tilde{\boldsymbol{z}}_m \;=\; \left[I_{LP} - \frac{1}{\gamma_m}\left(\vec{\omega}_m \otimes I_P\right)\left(\vec{\omega}_m^T \otimes I_P\right)\right]\boldsymbol{\mu}_{\tilde{z}_m|x} + \frac{1}{\gamma_m}\left(\vec{\omega}_m \otimes I_P\right)\vec{y}_m$$

Applying the properties of the Kronecker tensor product listed in the beginning of this chapter yields:

$$\tilde{\boldsymbol{z}}_m \;=\; \left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes I_P\right]\boldsymbol{\mu}_{\tilde{z}_m|x} + \frac{1}{\gamma_m}\left(\vec{\omega}_m \otimes I_P\right)\vec{y}_m,$$

which is applicable even if the model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is *not* assumed. Similarly, from Equation (4.68) we have:

$$\boldsymbol{\mu}_{\tilde{z}_m|x} \;=\; \boldsymbol{\mu}_{\tilde{z}_m} + \frac{1}{a_m}\left(I_L \otimes B_{m,1}\mathbf{s}\right)\left(\boldsymbol{x}_m - \left(I_L \otimes \mathbf{s}^T\right)\boldsymbol{\mu}_{\tilde{z}_m}\right),$$

where $a_m = \mathbf{s}^T B_{m,1}\mathbf{s}$. Substituting the latter expression into the former and collecting terms yields the super-pixel breakdown of the Form 1 MAP estimator under the noise free and special case assumptions:

$$\begin{aligned}\tilde{\boldsymbol{z}}_m \;&=\; \left[(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T) \otimes (I_P - \frac{1}{a_m}B_{m,1}\mathbf{s}\mathbf{s}^T)\right]\boldsymbol{\mu}_{\tilde{z}_m} \\ &+\; \frac{1}{a_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right]\boldsymbol{x}_m + \frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P)\vec{y}_m. \end{aligned} \tag{4.70}$$

Since this expression is free of matrix inverses, it is a candidate for real time implementation. It is expected that it will provide a better result than interpolation of $\boldsymbol{y}$ alone (without use of $\boldsymbol{x}$) while keeping complexity to a level suitable for real time implementation. This expectation is based on the assumption that estimates of the covariance matrix are performed in parallel in an efficient manner.

4.12 Lagrange Multiplier Optimization

In this section, we present an alternate derivation of the Form 1 MAP estimate using the optimization technique of Lagrange multipliers. The derivation applies only to the case of no noise present in the observed data. Rather than an unconstrained minimization of (4.29), we wish to minimize

$$C(\boldsymbol{z}) = \frac{1}{2}(\boldsymbol{z} - \boldsymbol{\mu}_{z|x})^T C_{z|x}^{-1}(\boldsymbol{z} - \boldsymbol{\mu}_{z|x}) \tag{4.71}$$

subject to the constraint that $\boldsymbol{y} = W\boldsymbol{z}$. Multiplying out the terms and keeping only those that are functions of $\boldsymbol{z}$, we arrive at an equivalent cost function

$$C(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T C_{z|x}^{-1}\boldsymbol{z} - \left(\boldsymbol{\mu}_{z|x}^T C_{z|x}^{-1}\right)\boldsymbol{z}. \tag{4.72}$$

Thus, we are faced with a minimization of a quadratic form with linear constraints. We can solve the problem using the method of Lagrangian multipliers [17]. The Lagrangian is formed as

$$L(\boldsymbol{z}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{z}^T C_{z|x}^{-1}\boldsymbol{z} - \left(\boldsymbol{\mu}_{z|x}^T C_{z|x}^{-1}\right)\boldsymbol{z} + \boldsymbol{\lambda}^T(W\boldsymbol{z} - \boldsymbol{y}), \tag{4.73}$$

where $\boldsymbol{\lambda}$ is an $MP \times 1$ vector of Lagrangian multipliers. The gradient with respect to $\boldsymbol{z}$ is given by

$$\nabla_{\boldsymbol{z}} L(\boldsymbol{z}, \boldsymbol{\lambda}) = C_{z|x}^{-1}\boldsymbol{z} - (C_{z|x}^{-1})^T\boldsymbol{\mu}_{z|x} + W^T\boldsymbol{\lambda}. \tag{4.74}$$

Setting this equal to zero and solving for $\boldsymbol{z}$ yields

$$\hat{\boldsymbol{z}} = \boldsymbol{\mu}_{z|x} - C_{z|x}W^T\boldsymbol{\lambda}. \tag{4.75}$$

Now let us find the $\boldsymbol{\lambda}$ that allows our solution to meet the linear constraint. To do so we impose the linear constraint on (4.75) and solve for $\boldsymbol{\lambda}$. This gives us

$$\boldsymbol{y} = W\hat{\boldsymbol{z}} = W\left(\boldsymbol{\mu}_{z|x} - C_{z|x}W^T\boldsymbol{\lambda}\right). \tag{4.76}$$

Solving for $\boldsymbol{\lambda}$ we get

$$\boldsymbol{\lambda} = \left(WC_{z|x}W^T\right)^{-1}\left(W\boldsymbol{\mu}_{z|x} - \boldsymbol{y}\right). \tag{4.77}$$

Plugging this into (4.75) yields our final solution

$$\hat{\boldsymbol{z}} = \boldsymbol{\mu}_{z|x} - C_{z|x}W^T\left[\left(WC_{z|x}W^T\right)^{-1}\left(W\boldsymbol{\mu}_{z|x} - \boldsymbol{y}\right)\right]. \tag{4.78}$$

This result matches Equation (4.57) when the covariance of the noise is set to zero.

4.13 Generalization to Multispectral Images

Up to this point, the $N \times 1$ vector $\boldsymbol{x}$ has represented an $n_v \times n_h$ (i.e. panchromatic) image. We now generalize the most important equations (those required for computer implementation) to the case where $\boldsymbol{x}$ has additional spectral content. In particular, we assume that $\boldsymbol{x}$ is an $n_v \times n_h \times \nu$ data cube written as an $N\nu \times 1$ vector. In reference [18], the quantity ν is denoted by Q. We chooses a different variable name in order to avoid confusion with the $NP \times NP$ permutation matrix Q.

We adopt the following ordering and notation for the $N\nu \times 1$ multispectral vector $\boldsymbol{x}$:

$$\boldsymbol{x} \;=\; \left[x^{(1)}; x^{(2)}; \ldots; x^{(\nu)}\right],$$

where $x^{(k)}$ for $1 \le k \le \nu$ is an $N \times 1$ vector whose entries are denoted by

$$x^{(k)} \;=\; \left(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)}\right)^T.$$

Thus, $x^{(k)}$ contains the spatial content of $\boldsymbol{x}$ corresponding to multispectral band k. When thought of as an $n_v \times n_h$ matrix, the entries of $x^{(k)}$ are stored in column order.

Our existing MATLAB code implements the Form 1 MAP algorithm through the use of Equation (4.67). Fortunately, the generalization to multispectral data cubes does not alter this equation. However, under the assumption $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$, the matrix S and the vector $\boldsymbol{\eta}$ take on new meaning, and this has an impact on how the various terms and factors of Equation (4.67) are computed. Also affected is the least squares approach presented in Section 4.6 for estimating the spectral response matrix from $\boldsymbol{x}$ and $\boldsymbol{y}$.

The matrix S changes from an $N \times NP$ matrix to an $N\nu \times NP$ matrix and $\boldsymbol{\eta}$ changes from an $N \times 1$ vector to an $N\nu \times 1$ vector. In accordance with S, the $P \times 1$ spectral response vector $\mathbf{s}$ becomes a $P \times \nu$ matrix whose entries are nonnegative and whose column sums are unity.

For $k = 1, 2, \ldots, \nu$, let $s^{(k)}$ denote the k^{th} column of the $P \times \nu$ matrix $\mathbf{s}$. Define the $N \times NP$ matrix $\widetilde{S}^{(k)}$ by:

$$\widetilde{S}^{(k)} \;=\; \bigoplus_{n=1}^{N} [s^{(k)}]^T \tag{4.79}$$

$$\;=\; I_N \otimes [s^{(k)}]^T. \tag{4.80}$$

Now define the $N\nu \times NP$ matrix $\widetilde{S}$ by vertical concatenation of the $\widetilde{S}^{(k)}$:

$$\widetilde{S} \;=\; \left[\widetilde{S}^{(1)}; \widetilde{S}^{(2)}; \ldots; \widetilde{S}^{(\nu)}\right]. \tag{4.81}$$

The $N\nu \times NP$ matrix S for the multispectral case is given by $S = \widetilde{S}Q$.

One consequence of changing $\mathbf{s}$ to a $P \times \nu$ matrix is that the algorithm for its estimation as given in Section 4.6 changes slightly. We define $\bar{x}_m^{(k)}$ analogous to Equation (4.40) as follows:

$$\bar{x}_m^{(k)} = \frac{1}{L} \sum_{j=1}^{L} x_{q((m-1)L+j)}^{(k)}.$$

In analogy to Equation (4.41)we set:

$$\bar{x}_m^{(k)} \;=\; \left\langle \vec{y}_m, s^{(k)} \right\rangle \quad (1 \le m \le M).$$

In matrix form (analogous to Equation (4.42)) this becomes:

$$Y s^{(k)} \;=\; \bar{x}^{(k)},$$

where $\bar{x}^{(k)} = (\bar{x}_1^{(k)}, \bar{x}_2^{(k)}, \ldots, \bar{x}_M^{(k)})$ and Y is the same as before. Thus, there are now ν systems of equations to solve in the least squares sense. Each system is subject to the constraints that $s^{(k)}$ contain nonnegative entries that sum to unity. The MATLAB function `LSQLIN` may still be used, but now it appears within a loop over k.

Equation (4.67), which we used for programming the Form 1 MAP algorithm, ultimately depends on the $P \times 1$ conditional hyper-pixel means $\mu_{\tilde{z}_i|x}$ and the $P \times P$ conditional covariance matrices $G_{m,j}$ (i.e. the j^{th} $P \times P$ block of $C_{\tilde{z}|x}$ corresponding to the m^{th} super-pixel). Under the assumption $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$, these are estimated by their (unconditional) counterparts $\mu_{\tilde{z}_i}$ and $B_{m,j}$ through use of Equations (4.48) and (4.53). However, these equations were derived under the assumption of $\boldsymbol{x}$ being panchromatic and do not apply to the multispectral case.

To derive the generalized versions of Equations (4.48) and (4.53), we start with the more fundamental Equations (4.26) and (4.27). Let D denote the $N\nu \times N\nu$ matrix in square brackets that appears in both of these equations:

$$D \;=\; \widetilde{S} C_{\tilde{z}} \widetilde{S}^T + C_\eta,$$

where $C_\eta = \sigma_\eta^2 \, I_{N\nu}$. Our first problem will be to invert D.

From Equation (4.81), D is naturally partitioned into ν^2 $N \times N$ blocks, where block (i, j) is defined by:

$$D_{i,j} \;=\; \widetilde{S}^{(i)} C_{\tilde{z}} [\widetilde{S}^{(j)}]^T + \delta_{i,j} \, \sigma_\eta^2 \, I_N,$$

and $\delta_{i,j}$ denotes the usual Kronecker delta function (1 if $i = j$ and 0 otherwise).

Equations (4.80) and (4.46) yield the following expression for $D_{i,j}$:

$$\begin{aligned} D_{i,j} &= \bigoplus_{n=1}^{N} d_n(i,j), \\ d_n(i,j) &= [s^{(i)}]^T B_n s^{(j)} + \delta_{i,j} \, \sigma_\eta^2. \end{aligned}$$

It follows that for each $n = 1, 2, \ldots, N$, the $\nu \times \nu$ matrix d_n is given by:

$$d_n \;=\; \mathbf{s}^T B_n \mathbf{s} + \sigma_\eta^2 \; I_\nu. \tag{4.82}$$

Note that these matrices reduce to the scalars d_i of Section 4.7 for the case of $\nu = 1$.

Denote the $N\nu \times N\nu$ inverse of D by U and partition it in a manner consistent with D, using $U_{i,j}$ to denote its (i,j) $N \times N$ block. We now *assume* that the zero/nonzero structure of U is identical to that of D. Our assumption will be justified if the resulting nonzero entries of U are uniquely determined. Accordingly, assume $U_{i,j}$ is given in terms of unknown scalars $u_n(i,j)$ by:

$$U_{i,j} \;=\; \bigoplus_{n=1}^{N} u_n(i,j),$$

and compute the $u_n(i,j)$ such that $DU = I_{N\nu}$. The matrix equation $DU = I_{N\nu}$ is equivalent to:

$$\sum_{k=1}^{\nu} D_{i,k} U_{k,j} \;=\; \delta_{i,j} \; I_N,$$

where i and j run over $1, 2, \ldots, \nu$. Given the assumed form of $U_{i,j}$, this becomes:

$$\bigoplus_{n=1}^{N} \left[\sum_{k=1}^{\nu} d_n(i,k) u_n(k,j) \right] \;=\; \delta_{i,j} \; I_N.$$

This however is equivalent to

$$\sum_{k=1}^{\nu} d_n(i,k) u_n(k,j) \;=\; \delta_{i,j},$$

as n runs over $1, 2, \ldots, N$, which is just another way of asserting that the $\nu \times \nu$ matrices d_n and u_n are inverses of each other. Thus, the elements of u_n are uniquely determined by $u_n = d_n^{-1}$ and the inversion of D reduces to the inversion of the $\nu \times \nu$ matrices $d_1, d_2, \ldots, d_N$. It is seen from Equation (4.82) that the condition of d_n will improve as σ_η^2 increases. Just as the noise variance σ_n^2 may be considered a diagonal load to improve the condition of $\widetilde{W} C_{\tilde{z}|x} \widetilde{W}^T$, the noise variance σ_η^2 may be considered a diagonal load to improve the condition of $\widetilde{S} C_{\tilde{z}} \widetilde{S}^T$.

Continuing with the generalization of Equations (4.48) and (4.53), we derive expressions in terms of the scalars $u_n(i,j)$, which we now know how to compute. We proceed by

dissecting the factor $\widetilde{S}^T D^{-1}\widetilde{S}$ appearing in Equation (4.27):

$$\begin{aligned}\widetilde{S}^T D^{-1}\widetilde{S} &= \sum_{j=1}^{\nu}\sum_{i=1}^{\nu}[\widetilde{S}^{(i)}]^T U_{i,j}\widetilde{S}^{(j)} \\ &= \sum_{j=1}^{\nu}\sum_{i=1}^{\nu}\left(\bigoplus_{n=1}^{N} s^{(i)}u_n(i,j)[s^{(j)}]^T\right) \\ &= \bigoplus_{n=1}^{N}\left(\sum_{j=1}^{\nu}\sum_{i=1}^{\nu} u_n(i,j)s^{(i)}[s^{(j)}]^T\right).\end{aligned}$$

Substituting this factor into Equation (4.27) yields:

$$\begin{aligned}C_{\tilde{z}|x} &= C_{\tilde{z}} - C_{\tilde{z}}\left(\widetilde{S}^T D^{-1}\widetilde{S}\right)C_{\tilde{z}} \\ &= \bigoplus_{n=1}^{N}\left(B_n - B_n\sum_{j=1}^{\nu}\sum_{i=1}^{\nu} u_n(i,j)s^{(i)}[s^{(j)}]^T B_n\right) \\ &= \bigoplus_{n=1}^{N}\left(B_n - \sum_{i=1}^{\nu}\sum_{j=1}^{\nu} u_n(i,j)B_n s^{(i)}[B_n s^{(j)}]^T\right).\end{aligned}$$

It follows that the conditional covariance matrices $G_{m,j}$ representing the j^{th} $P\times P$ matrix within the m^{th} super-pixel of $C_{\tilde{z}|x}$ is given by:

$$G_{m,j} = B_{m,j} - \sum_{i=1}^{\nu}\sum_{k=1}^{\nu}\left(u_n(i,k)B_{m,j}s^{(i)}\right)[B_{m,j}s^{(k)}]^T, \tag{4.83}$$

which generalizes Equation (4.53).

We conclude with deriving a generalization of Equation (4.48). Dissecting the factor $C_{\tilde{z}}\widetilde{S}^T D^{-1}$ of Equation (4.26), we have:

$$C_{\tilde{z}}\widetilde{S}^T D^{-1} = \left(\bigoplus_{n=1}^{N}\sum_{i=1}^{\nu} u_n(i,1)B_n s^{(n)}, \bigoplus_{n=1}^{N}\sum_{i=1}^{\nu} u_n(i,2)B_n s^{(n)}, \ldots, \bigoplus_{n=1}^{N}\sum_{i=1}^{\nu} u_n(i,\nu)B_n s^{(n)}\right).$$

Similarly, the factor $\boldsymbol{x} - \widetilde{S}\boldsymbol{\mu}_{\tilde{z}}$ of Equation (4.26) is:

$$\begin{aligned}\boldsymbol{x} - \widetilde{S}\boldsymbol{\mu}_{\tilde{z}} &= \left(x^{(1)} - \widetilde{S}^{(1)}\boldsymbol{\mu}_{\tilde{z}}; x^{(2)} - \widetilde{S}^{(2)}\boldsymbol{\mu}_{\tilde{z}}; \ldots; x^{(\nu)} - \widetilde{S}^{(\nu)}\boldsymbol{\mu}_{\tilde{z}}\right) \\ x^{(k)} - \widetilde{S}^{(k)}\boldsymbol{\mu}_{\tilde{z}} &= \left(x_1^{(k)} - [s^{(k)}]^T\mu_{\tilde{z}_1}, x_2^{(k)} - [s^{(k)}]^T\mu_{\tilde{z}_2}, \ldots, x_N^{(k)} - [s^{(k)}]^T\mu_{\tilde{z}_N}\right)^T.\end{aligned}$$

Putting the factors together yields:

$$C_{\tilde{z}}\widetilde{S}^T D^{-1}\left(\boldsymbol{x}-\widetilde{S}\boldsymbol{\mu}_{\tilde{z}}\right) = \sum_{k=1}^{\nu}\left[\left(\bigoplus_{n=1}^{N}\sum_{i=1}^{\nu}u_n(i,k)B_n s^{(i)}\right)\begin{pmatrix} x_1^{(k)}-[s^{(k)}]^T\mu_{\tilde{z}_1} \\ x_2^{(k)}-[s^{(k)}]^T\mu_{\tilde{z}_2} \\ \vdots \\ x_N^{(k)}-[s^{(k)}]^T\mu_{\tilde{z}_N}\end{pmatrix}\right]$$
$$= \bigoplus_{n=1}^{N}\left[\sum_{i=1}^{\nu}\sum_{k=1}^{\nu}\left(x_n^{(k)}-[s^{(k)}]^T\mu_{\tilde{z}_n}\right)u_n(i,k)B_n s^{(i)}\right].$$

Therefore, through the use of Equation (4.26), the n^{th} $P\times 1$ conditional mean high resolution hyper-pixel is given in terms of the corresponding (unconditional) mean high resolution hyper-pixel by:

$$\mu_{\tilde{z}_n|x} = \mu_{\tilde{z}_n}+\sum_{i=1}^{\nu}\sum_{k=1}^{\nu}\left(x_n^{(k)}-[s^{(k)}]^T\mu_{\tilde{z}_n}\right)u_n(i,k)B_n s^{(i)}, \tag{4.84}$$

which generalizes Equation (4.48). Equations (4.82), (4.83), and (4.84) are some of the more important equations relevant to programming a generalized MAP algorithm that applies to both multispectral and panchromatic inputs.

4.14 Estimation of Conditional Parameters

We now discuss our approach to estimating conditional statistical parameters $C_{z|x}$ and $\boldsymbol{\mu}_{z|x}$ under the assumption that the linear model $\boldsymbol{x}=S\boldsymbol{z}+\boldsymbol{\eta}$ does *not* hold. In this section and in Section 4.13, we generalize the single band panchromatic array $\boldsymbol{x}$ to a multispectral array with ν bands. Thus,

$$\boldsymbol{x} = (\vec{x}_1^T,\vec{x}_2^T,\ldots,\vec{x}_N^T)^T,$$

where each $\vec{x}_n$ is a $\nu\times 1$ vector (for a panchromatic image, $\nu=1$).

Estimating the full $NP\times NP$ covariance matrix $C_{z|x}$, used in Equation (4.28), is impractical for a typical size hyperspectral image. To make the problem manageable, constraints on the form of this covariance are required to bring down the number of statistical parameters to be estimated. While there may be numerous ways to accomplish this, we believe that a reasonable approach is to model the unknown high-resolution hyper-pixels as conditionally independent spatially, yielding

$$P_r(\boldsymbol{z}|\boldsymbol{x})=\prod_{n=1}^{N}P_r(\vec{z}_n|\vec{x}_n), \tag{4.85}$$

which is the multispectral version of Level 1 conditional independence presented in Section 4.9. Writing out the individual conditional PDFs yields

$$P_r(\boldsymbol{z}|\boldsymbol{x}) = \prod_{n=1}^{N} \frac{1}{\sqrt{(2\pi)^P |C_{z_n|x_n}|}} \exp\left\{-\frac{1}{2}(\vec{z}_n - \boldsymbol{\mu}_{z_n|x_n})^T C_{z_n|x_n}^{-1}(\vec{z}_n - \boldsymbol{\mu}_{z_n|x_n})\right\}, \tag{4.86}$$

where $\boldsymbol{\mu}_{z_n|x_n} = \mathrm{E}\{\vec{z}_n|\vec{x}_n\}$ and $C_{z_n|x_n}$ is the $P \times P$ covariance matrix for $\vec{z}_n$ given $\vec{x}_n$. The relationship between the individual hyper-pixel statistical parameters and the global statistical parameters in Equation (4.28) is given by

$$C_{z|x} = \begin{bmatrix} C_{z_1|x_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & C_{z_2|x_2} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & C_{z_N|x_N} \end{bmatrix} = \bigoplus_{n=1}^{N} C_{z_n|x_n}, \tag{4.87}$$

and

$$\boldsymbol{\mu}_{z|x} = \left[\boldsymbol{\mu}_{z_1|x_1}^T, \boldsymbol{\mu}_{z_2|x_2}^T, \ldots, \boldsymbol{\mu}_{z_N|x_N}^T\right]^T. \tag{4.88}$$

Furthermore, applying the results in Equations (4.21) and (4.22) on the hyper-pixels yields

$$\boldsymbol{\mu}_{z_n|x_n} = \mathrm{E}\{\vec{z}_n\} + C_{z_n,x_n} C_{x_n}^{-1} \left[\vec{x}_n - \mathrm{E}\{\vec{x}_n\}\right] \tag{4.89}$$

and

$$C_{z_n|x_n} = C_{z_n} - C_{z_n,x_n} C_{x_n}^{-1} C_{z_n,x_n}^T. \tag{4.90}$$

Note that the covariance matrix of the joint random vector, $\boldsymbol{\psi}_n = \left[\vec{x}_n^T, \vec{z}_n^T\right]^T$, is related to the cross-covariance matrices in Equations (4.89) and (4.90) as follows

$$C_{\psi_n} = \begin{bmatrix} C_{x_n,x_n} & C_{z_n,x_n}^T \\ C_{z_n,x_n} & C_{z_n,z_n} \end{bmatrix}. \tag{4.91}$$

With the simplifying assumption of conditional independence, $C_{z|x}$ is block diagonal, reducing the number of statistical parameters from $(NP)^2$ to NP^2. That is, the problem reduces to estimating the hyper-pixel conditional statistics in Equations (4.89) and (4.90). The number of statistical parameters may be further reduced by assuming some or all hyper-pixels as having the same conditional covariance. If, for example, all hyper-pixels have the same covariance, then we have only P^2 statistical parameters in the conditional covariance matrix to estimate. *An area of future work might be to explore other forms for the conditional covariance matrix* (implying other assumptions regarding the nature of the high-resolution hyper-pixels).

To estimate $\mathrm{E}\{\vec{z}_n\}$, we propose using the spatially interpolated observed hyperspectral imagery, denoted $\hat{\boldsymbol{\mu}}_{z_n}$. We have found that spline interpolation tends to yield the best results here. To estimate $\mathrm{E}\{\vec{x}_n\}$, we use a spatially smoothed version of the band or bands in $\boldsymbol{x}$. We have observed that the best results are obtained when the smoothing is done to

mimic the way in which $\hat{\boldsymbol{\mu}}_{\vec{z}_n}$ relates to $\vec{z}_n$. That is, we degrade $\vec{x}_n$ with a PSF and down-sampling factor similar to that defined by W. This degraded image is interpolated using spline interpolation to produce the estimate of $\mathrm{E}\{\vec{x}_n\}$, which we will denote $\hat{\boldsymbol{\mu}}_{x_n}$, for all n. In this fashion it may be said that $\hat{\boldsymbol{\mu}}_{x_n}$ relates to $\vec{x}_n$ as $\hat{\boldsymbol{\mu}}_{z_n}$ relates to $\vec{z}_n$. These estimates tend to track the non-stationarity in the mean exhibited by most natural images [21].

The joint covariance in Equation (4.91) must also be estimated in order to get the conditional statistics in (4.89) and (4.90). In most cases it will not be possible to obtain statistically similar data at the high resolution required. Thus, we attempt to estimate the required joint covariance from the observed imagery. To do so, we will estimate a joint covariance at the lower resolution of the observed hyperspectral imagery and apply it at the higher resolution. While the joint statistics may differ at different resolutions, it is hoped that there is sufficient symmetry of spatial scale in the statistical parameters to provide a useful result. In particular, we artificially degrade the spatial resolution and size of $\boldsymbol{x}$ to match that of $\boldsymbol{y}$. Let this degraded image be denoted $\tilde{\mathbf{x}} = \left[\tilde{\boldsymbol{x}}_1^T, \tilde{\boldsymbol{x}}_2^T, ..., \tilde{\boldsymbol{x}}_M^T\right]^T$. The local means at this resolution, obtained using the same method applied to the original resolution, are removed from $\tilde{\mathbf{x}}$ and $\boldsymbol{y}$. Now, the joint covariance information is estimated. One relatively simple approach seeks a single global covariance using a sample covariance estimate. This covariance is used as an estimate of C_{ψ_n} for all n.

However, in order to more fully exploit the information in $\boldsymbol{x}$, we wish to capture the changing joint covariance as the spectral content in the scene varies spatially. Since it is impractical to estimate a joint covariance for each hyper-pixel, we use a simple clustering approach based on vector quantization. To begin, we form joint vector $\tilde{\boldsymbol{\psi}}_m = \left[\tilde{\boldsymbol{x}}_m^T, \boldsymbol{y}_m^T\right]^T \in \mathbb{R}^{\nu+P}$ for $m = 1, 2, \ldots, M$, where $\mathbb{R}^{\nu+P}$ represents the $\nu + P$ dimensional real space. These vectors are grouped into K clusters (or classes) using the Linde-Buzo-Gray (LBG) algorithm [22]. The cluster centroids define the Voronoi partitions of the spectral space. That is, a given vector in $\mathcal{R}^{\nu+\mathcal{P}}$ space is assigned to class k if it lies closest, in a Euclidean sense, to the centroid of cluster k. For each cluster, the sample joint covariance is computed using $\tilde{\boldsymbol{\psi}}_m$ for $m \in \Omega_k$, where Ω_k is the set of all indices corresponding to Voronoi partition k. To estimate the joint covariance, C_{ψ_n}, we assign $\hat{\boldsymbol{\psi}}_n = \left[\vec{x}_n^T, \hat{\boldsymbol{\mu}}_{z_n}^T\right]^T$ to a partition and let the covariance for this high-resolution spatial position n be the corresponding cluster covariance. With these covariance estimates in hand, the final MAP estimate can be formed. In some cases we have observed improvement in performance if we form an estimate of the conditional mean in Equation (4.89), $\hat{\boldsymbol{\mu}}_{z_n|x_n}$, and then re-classify the image using $\hat{\boldsymbol{\psi}}_n = [\vec{x}_n^T, \hat{\boldsymbol{\mu}}_{z_n|x_n}^T]^T$ for the purpose of assigning cluster covariances to each high-resolution position.

Chapter 5

MAP Algorithm Performance

Focusing on the use of high-resolution panchromatic (single band) data to enhance hyperspectral imagery, we show that the MAP estimator produces an estimate with enhanced sub-pixel spectral content that is evident not only in the first principal component image, but in lower components as well. The relationship of the proposed method to some prior methods is discussed in Section 5.1. Experimental results are presented in Section 5.2. Finally, conclusions are given in Section 5.3.

5.1 Relationship to Other Approaches

It is interesting to explore the relationship between the proposed MAP estimate and some previously proposed approaches. Consider that if one neglects the observation model for $\boldsymbol{y}$ in the MAP formulation, the resulting cost function would simply be the second term in Equation (4.29). This would lead to an estimate that is the conditional mean in Equation (4.89). This result is essentially the same as that derived by Nishii *et al.* [1] for Landsat Thematic Mapper thermal band estimation, given specific choices for the estimates of $\mathrm{E}\{\vec{z}_n\}$ and $\mathrm{E}\{\vec{x}_n\}$. In particular, one must use zero-order-hold (ZOH) interpolation (i.e., pixel replication) on each band of the observed hyperspectral image to estimate $\mathrm{E}\{\vec{z}_n\}$ and average the auxiliary image pixels within the span of each low-resolution hyper-pixel to estimate $\mathrm{E}\{\vec{x}_n\}$. Using these estimates for $\mathrm{E}\{\vec{z}_n\}$ and $\mathrm{E}\{\vec{x}_n\}$ guarantees that the average of the estimated hyper-pixels within the span of a low-resolution hyper-pixel will be equal to the low-resolution hyper-pixel. Nishii *et al.* [1] explore the use of both local and global covariances. Thus, in comparison to their method, the proposed MAP framework is novel in how it explicitly incorporates an arbitrary system PSF and in how it allows for various statistical models and estimates of the statistical parameters. We will use the method of Nishii *et al.* [1] as one of our performance benchmarks for comparison in Section 5.2.

Another method used as a benchmark is the estimator proposed by Price [2, 7]. This method was designed to combine multispectral imagery with a panchromatic auxiliary image

(i.e., $\nu = 1$). For bands strongly correlated with the panchromatic sensor, the estimate is based on a linear mapping of the panchromatic image, yielding

$$\hat{z}_{p,n} = a_p x_{1,n} + b_p, \tag{5.1}$$

for $n = 1, 2, ..., N$ and $p = 1, 2, ..., P$. This estimate is then scaled so that the average of the high-resolution hyper-pixels within the span of a low-resolution hyper-pixel is equal to the low-resolution hyper-pixel. The coefficients, a_p and b_p are estimated with least-squares regression using low-resolution hyperspectral band p and a degraded version of the panchromatic image (degraded to match the spatial resolution of the low-resolution hyperspectral band). For weakly correlated bands, a look-up-table (LUT) method is employed [2, 7]. The LUT is created based on the relationship between the low-resolution pixel values in a given band and the corresponding pixel values in the degraded panchromatic image. Once the LUT is generated it is applied to the full-resolution panchromatic image to form a high resolution estimate of the desired band. As before, this estimate is scaled so that the average of the high-resolution hyper-pixels within the span of a low-resolution hyper-pixel is equal to the low-resolution hyper-pixel.

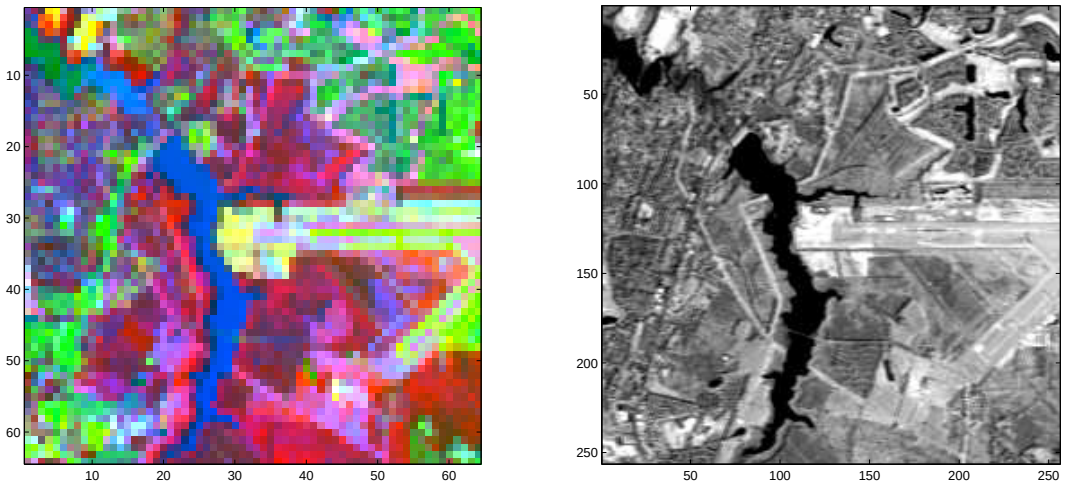
5.2 Experimental Results

In this section, we present a number of experimental results in order to demonstrate the efficacy of the proposed estimator in comparison to the benchmark techniques. Simulated data are used here to allow for quantitative performance analysis. The details of the data set are provided in Section 5.2.1. In Sections 5.2.2 and 5.2.3, quantitative error analysis is presented in the spectral band space and in the principal component space, respectively. Finally, noise analysis is presented in Section 5.2.4.

5.2.1 Simulated Data

The simulated data sets are derived from a hyperspectral image collected by the Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) sensor [23]. AVIRIS is a scanning dispersive hyperspectral imaging sensor that flies on the NASA ER-2 aircraft at approximately 20 km above sea level with a spatial resolution of approximately 6 m per pixel. The sensor collects 224 contiguous spectral bands in the range of 0.4 to 2.5 μm. The specific scene used has been collected over Yorktown Virginia (Flight F980703T01, Run 02, ID 1828000ST23).

A 256×256 portion of the scene is used as the true high spatial resolution hyperspectral image $\boldsymbol{z}$. This is artificially degraded to form $\boldsymbol{y}$. A simple rectangular detector model is used for the system PSF [24]. In particular, the PSF is a 4×4 kernel with equal weights of 1/16. The image is subsampled by a factor of 4 in both spatial dimensions. This PSF model leads to a simple structure in the matrix W described in Section 4.3. This structure, combined with the spatial conditional independence assumption, allows us to process each



(a) (b)

Figure 5.1: Simulated observed images derived from AVIRIS data. (a) False color image of principal components one, two, and three of the low spatial resolution hyperspectral data. (b) High spatial resolution panchromatic image.

low-resolution hyper-pixel to form a corresponding 4×4 set of hyper-pixels independently (after the mean estimates are formed using interpolation). For imagery in the mid- and long-wave infrared, diffraction effects tend to become larger and can be added to the observation model [24]. The associated high resolution sensor in this case is modeled as a panchromatic broadband imager ($\nu = 1$). These data are formed by averaging the 224 AVIRIS bands at the original resolution. A false color image, formed by mapping the first three principal components of the low-resolution hyperspectral data to red, green and blue, respectively, is shown in Figure 5.1(a). The simulated broadband image is shown in Figure 5.1(b).

The eigenvalue (variance) associated with each principal component of the low-resolution hyperspectral data is plotted in Figure 5.2. This clearly indicates that the vast majority of signal power is contained in the leading components. For example, after 20 components, the eigenvalue has dropped by approximately five orders of magnitude from the top component. In order to reduce the computational burden, we process the imagery in the PCA space in the top twenty dimensions. The lower 204 dimensions are processed using spline interpolation. The processed components are then transformed back to the original spectral space. Note that due to the nature of the PCA transformation, the estimation algorithm applies identically in the principal component space as shown in Section 4.10.

5.2.2 Spectral Space Performance Analysis

To quantitatively assess the performance of the MAP estimator, we compare the estimates with the "true" hyperspectral image (the original resolution AVIRIS image). Our image fidelity metric is signal-to-noise ratio (SNR), where "noise" here refers to estimation error.

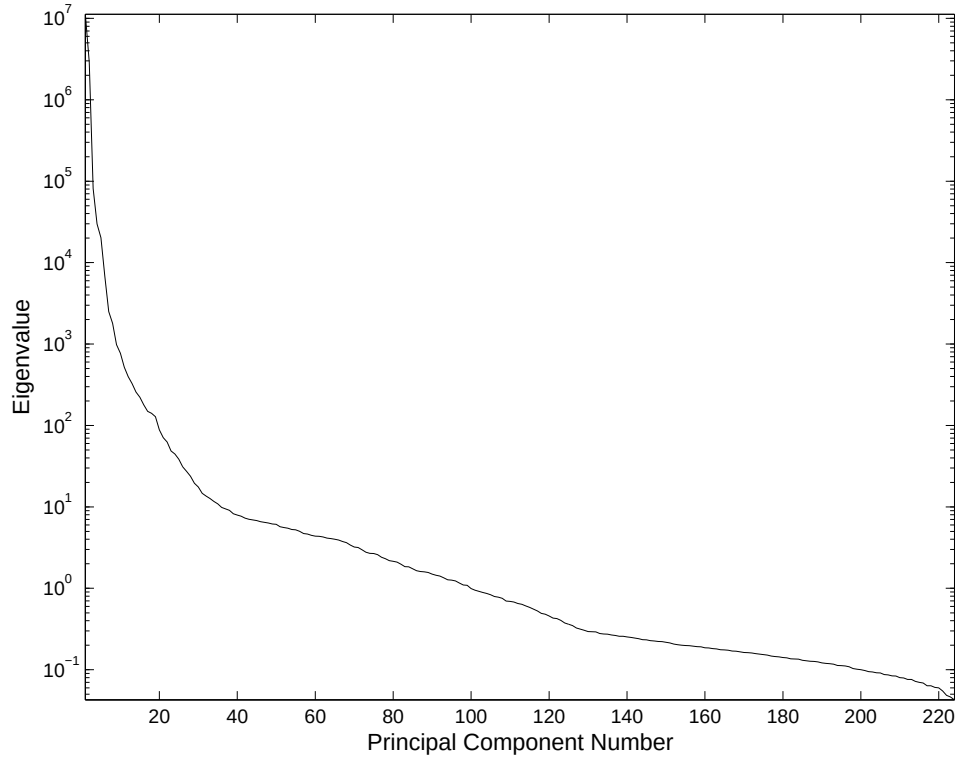


Figure 5.2: Eigenvalue versus component number for the low-resolution hyperspectral image.

This metric is computed as the sample variance of the "desired" image divided by the mean squared error (MSE). Scaling the reciprocal of the MSE by the variance of the desired image is helpful in allowing one to compare performance between bands with significantly different signal powers. This is particularly useful in principal component space, where power in the bands can vary by orders of magnitude.

The SNR versus wavelength is shown in Figure 5.3 for the MAP estimator ($K = 16$)[1], the method of Nishii *et al.* (with global covariance statistics) [1], Price's method [2], and spline interpolation. For Price's method, both the linear model and LUT approach are used and the best of the two SNRs for each band is reported. Here no noise is introduced to either the low-resolution hyperspectral imagery or the panchromatic imagery. The effects of noise are studied in Section 5.2.4. Note that significant improvement over spline interpolation is obtained in many bands with all the techniques. Not surprisingly, the bands with the highest correlation with the panchromatic image tend to have the highest SNRs. The spectral band estimates with very low SNR are a result of the original data having very low signal power due to atmospheric absorption. It can be seen from Figure 5.3 that the MAP estimate with ($K = 16$) provides the highest SNRs for these data.

[1] K refers to the number of VQ clusters as described in Section 4.14

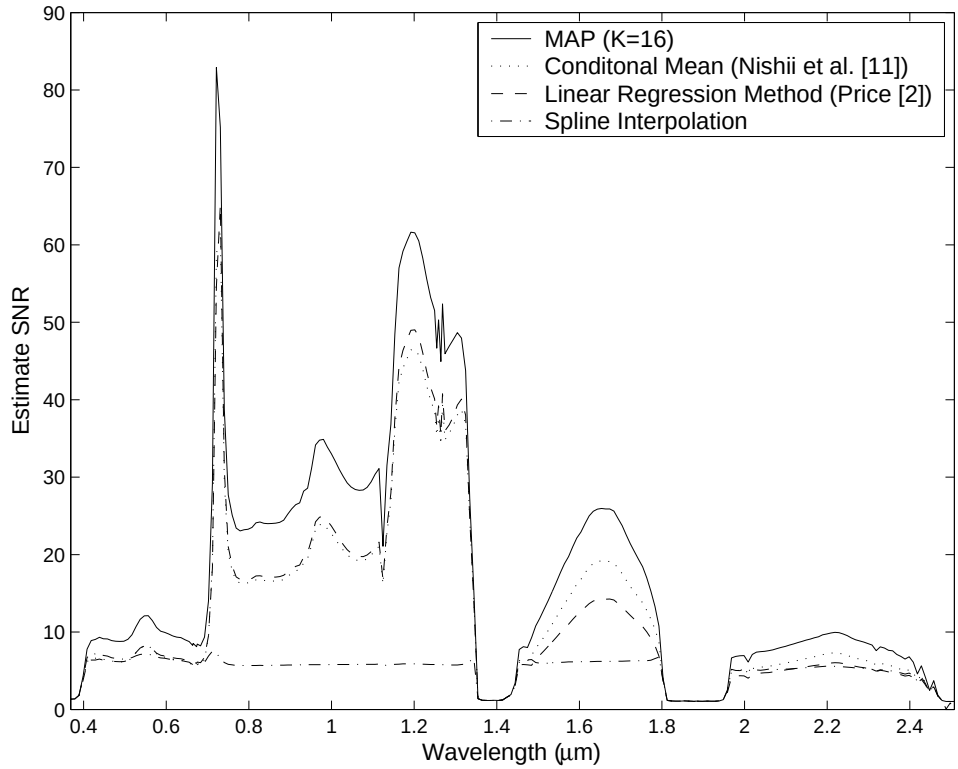


Figure 5.3: SNR versus AVIRIS band wavelength for the MAP estimator ($K = 16$), the method of Nishii *et al.* (with global covariance statistics) [1], Price's method [2], and spline interpolation.

5.2.3 Principal Component Space Performance Analysis

The spectral domain error analysis indicates that many spectral bands can be significantly enhanced with the use of the panchromatic imagery. However, it is insightful to examine the performance in the principal component space. Table 5.1 shows the SNR in the first 5 principal components for spline interpolation, the method of Nishii *et al.* (with global covariance statistics) [1], Price's method [2], and the MAP estimator for $K = 1$ and $K = 16$. Clearly the top principal component is dramatically improved by incorporating information from the panchromatic image. However, the lower components are far more difficult to enhance due to the weak correlation with the broadband image. The MAP estimator does provide a modest increase in SNR over spline interpolation for some of the lower components, while the benchmark techniques have lower SNRs than that obtained with spline interpolation. Note also that principal component substitution methods typically seek to enhance only the principal component, and do not enhance the lower components at all. Thus, we believe that *any* enhancement in these lower components is a promising result. The use of multispectral high-resolution imagery (rather than panchromatic) could provide the means to better improve these lower components. To focus on these lower components, Figure 5.4 shows the percentage SNR improvement over spline interpolation for the various estimators

Table 5.1: SNRs for estimates of the top 5 principal component images

Method	PC 1	PC 2	PC 3	PC 4	PC 5
Spline Interpolation	5.87	6.36	2.86	4.86	3.90
Conditional Mean (Nishii *et al.* [1])	26.92	5.69	2.48	4.21	3.62
Linear Regression Method (Price [2])	27.92	5.81	2.48	4.24	3.36
MAP ($K = 1$)	34.74	7.42	2.99	5.04	4.44
MAP ($K = 16$)	38.97	8.37	3.05	5.24	4.48

in components 2 through 20. Note that the MAP estimator with $K = 16$ generally shows the most improvement. We believe that the improvement seen with $K = 16$ versus $K = 1$ is because more correlation is present in the individual classes than exists globally.

False color images formed with the top three principal components mapped to red, green, and blue are shown in Figure 5.5. In particular, the true high resolution hyperspectral image components are shown in Figure 5.5(a). Spline interpolated components are shown in Figure 5.5(b). The estimate using Price's method is shown in Figure 5.5(c). Finally, the MAP estimate for $K = 16$ is shown in Figure 5.5(d). An enlargement of the upper right corner of the image is shown in Figure 5.6 for principal components two, three, and four. We believe that the MAP estimates generally appear sharper than the spline interpolated images and exhibit less prominent block artifacts than the estimates using Price's method (an observation consistent with the quantitative analysis).

The number and locations of the vector quantization code words that define the Voronoi partitions has a significant impact on the performance of the algorithm. Figure 5.7 shows the SNR versus the number of partitions for principal components 1, 2, and 3 for a larger portion (512×1536) of the AVIRIS scene. Note that for 8 and more partitions, the SNRs are markedly higher than with fewer partitions. However, the SNR does not appear to increase linearly with number of partitions, as one might expect. First, the performance of the LBG algorithm is somewhat sensitive to the initial starting point. The starting vectors are selected by uniformly subsampling the scene. Thus, with different numbers of partitions, different initial codewords are automatically selected. Secondly, the LBG algorithm is designed to find the most representative codewords (spectra). It is not designed to optimize the MAP algorithm performance. However, choosing a partitioning scheme to maximize the MAP algorithm performance is a daunting, possibly intractable, task. Furthermore, these results will depend on the specific scene used and the relative quantities of the spectra present. *Alternative methods for partitioning the observation space may be an interesting area for further research.*

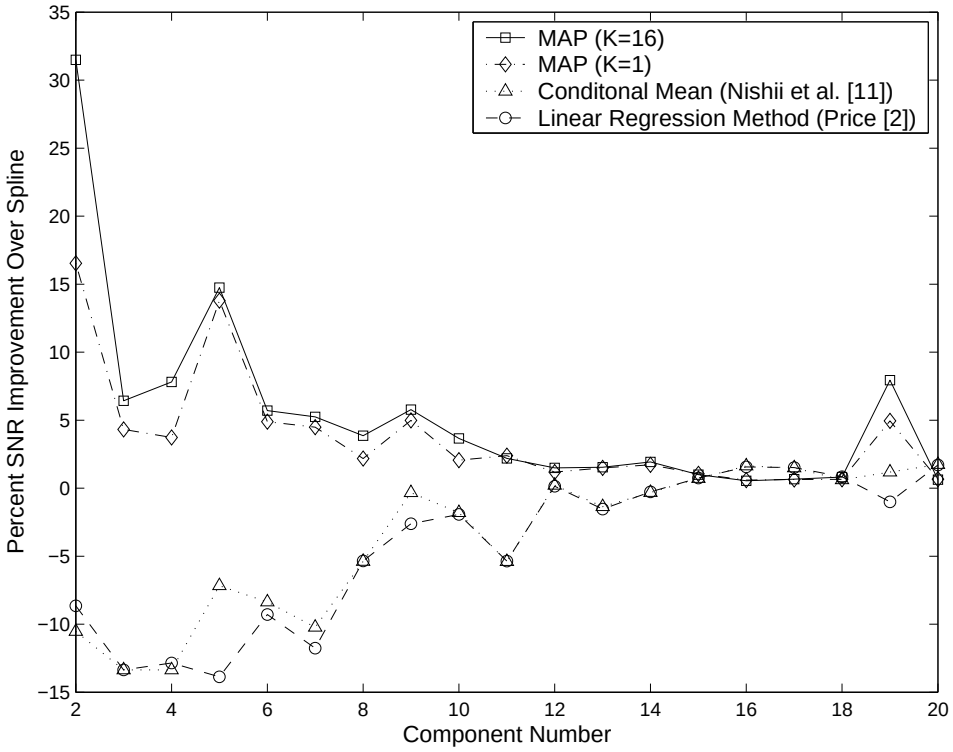


Figure 5.4: Percentage improvement in SNR over straight spline interpolation for estimates of principal components two through twenty.

5.2.4 Noise Analysis

In this section we consider how noise impacts the performance of the MAP estimator. First, we consider the impact of noise in the observed hyperspectral imagery, and then, we consider noise in the panchromatic imagery. The SNRs of the estimates of principal component two as a function of the average SNR of the observed low-resolution hyperspectral bands are shown in Figure 5.8. Here no noise in the panchromatic image is introduced. One curve shows the SNRs for the MAP estimates with $K = 16$ when zero noise variance is assumed. Another curve shows the SNRs when the correct noise variance is known and used.

The SNRs of the estimates of principal component two as a function of the panchromatic image SNR are shown in Figure 5.9. Here no noise in the hyperspectral image is introduced. Note that the spline interpolator does not depend on $\boldsymbol{x}$, and thus, is not affected by noise in the panchromatic image. Also note that when the SNR of the panchromatic image is low, little improvement is possible as correlation with the hyperspectral bands is reduced. With very low SNR panchromatic imagery, the correlation at the lower resolution is not indicative of the correlation at the higher resolution. This is because the noise at the lower resolution is reduced as the panchromatic image is artificially degraded to the lower resolution. The use of pre-filters for noise reduction could help to mitigate this effect.

5.2.5 Multispectral Auxiliary Sensor

We examined the performance of the MAP estimator using a multispectral auxiliary sensor. Figure 5.10 shows the SNR for the MAP estimator ($K = 16$) for a different 256×256 portion of the AVIRIS scene. We simulated a panchromatic sensor (averaging all AVIRIS bands), a visible/near infrared dual-band sensor (averaging the visible bands, and the remaining bands), and finally a six-band multispectral sensor with bands similar to those of the Landsat Thematic Mapper sensor (excluding the thermal band). Note that significant improvement is possible in some of the lower components, but the number of significantly improved principal components is approximately equal to the number of bands in the auxiliary sensor in these results. Modest improvement in many components is possible, however, as shown with the six-band auxiliary sensor results.

Figures 5.11 and 5.12 show some image results comparing panchromatic enhancement with multispectral enhancement. In particular, Figure 5.11(a) shows the raw low-resolution principal components 1, 2 and 3 in a false color composite. Figure 5.11(b) shows the same same components for the spline interpolated imagery. The high resolution components are shown in Figure 5.11(c). The MAP estimate with $K = 16$ and panchromatic sharpening is shown in Figure 5.11(d). The dual-band sharpened result is shown in Figure 5.11(e) and the six-band sharpened result is shown in Figure 5.11(f). The images are dominated by the principal component so that all look quite good. However, some improvement is seen with the dual-band and multispectral auxiliary sensor over the panchromatic sensor. Figure 5.12 shows the same set of results as Figure 5.11, except principal components 2, 3 and 4 are used in the false color composites. Here a more dramatic improvement can be seen when the dual-band and multispectral auxiliary sensor are used.

5.2.6 Matched Filter Analysis

To investigate the potential benefit of the processed imagery on spectral detection and land-cover classification, several matched filter experiments have been conducted. To begin, an area within the lake region of the AVIRIS imagery has been selected and the spectra in this area are averaged to form a target spectrum. A spectral matched filter using this target spectrum is applied to the high-resolution imagery, yielding the result shown in Figure 5.13(a). To cast this into a detection framework, this matched filter image has been thresholded to produce a truth mask, shown in Figure 5.13(b). Using the target spectrum, the matched filter result using the MAP estimate with $K = 16$ and panchromatic auxiliary sensor is obtained and shown in Figure 5.13(c). For comparison, the matched filter result using the spline interpolated imagery is shown in Figure 5.13(d). It is clear from the matched filter results that the matched filter using the MAP imagery does a much better job with the boundary of the lake region than the interpolated imagery.

A receiver operating characteristic (ROC) curve for the lake target is shown in Figure 5.14. This is based on the truth mask shown in Figure 5.13(b). Note that the area under the

ROC curve for the MAP estimate imagery is clearly higher than that for the interpolated imagery. This demonstrates a definite improvement in matched filter detection performance using the MAP imagery for this land-cover class. The most benefit has been observed for large landcover classes that are well represented in the vector quantization codebook and the corresponding class statistics. Rare spectra objects comprising only a small number of hyper-pixels in the scene may not be enhanced by this approach, relative to spline interpolation. One possible approach to enhancing rare spectra objects is to hand select some of the codewords that define the partitioning and providing sufficient training data for these rare spectra objects. If the objects are only present in a few hyper-pixels in the current imagery, additional imagery may be required showing the rare spectra objects amongst various backgrounds likely to be encountered in the real data (the mixing aspect is as important as the rare spectra itself).

5.3 Performance Summary and Conclusions

This research effort led to the development of a MAP estimation framework for estimating an enhanced resolution image using co-registered high-resolution imagery from another sensor. Here we have focused on the enhancement of a hyperspectral image using high-resolution panchromatic data. However, the estimation framework developed allows for any number of spectral bands in the primary and auxiliary image. We believe that the proposed technique is suitable for applications where some correlation exists between the auxiliary image and the image being enhanced. The results with AVIRIS imagery indicate that a number of methods do well enhancing the top principal component image, where strong global correlation exists with the panchromatic image. The lower component images are much more difficult to enhance. Notwithstanding this, we have demonstrated that the proposed estimator is capable of providing modest improvement in some of these lower components (something not seen with the benchmark techniques). The spatially varying statistical model (i.e., $K = 16$), using vector quantization, does provide some additional performance gain over global statistics (i.e., $K = 1$). We believe this is a result of exploiting the higher correlations present within the spectral classes (correlations that get "washed out" in the global statistics).

We believe that one of the merits of the proposed estimation framework is that it allows for an arbitrary linear observation model. Furthermore, the estimation framework opens up opportunities to improve upon these results with the use of more sophisticated statistical models and methods for estimating the statistical parameters for those models. For example, improved performance may be possible using a spatial-spectral model for the desired high-resolution image (i.e., not assuming the desired hyper-pixels are conditionally independent, given the auxiliary image).

The big question surrounding this research is: what benefit does this processing bring to the utility of the processed imagery? Clearly the imagery is enhanced for subjective human interpretation. *We believe that the MAP approach arguably represents the most theoretically*

sound method proposed to date for merging hyperspectral imagery with an auxiliary sensor. Thus, in any applications where panchromatic sharpening is used for human interpretation, our method could offer improved performance. In addition to the subject enhancement, we have demonstrated that some improvement, if only modest, is possible in several of the principal component images (not just the leading component). This in turn, may allow spectral based classifiers and detectors to yield superior results. We demonstrated that, with large landcover classes, boundaries are significantly enhanced. Admittedly, rare spectra objects may not be enhanced (without modification to the present algorithm). In light of this observation, change detection may be a particularly good application for the MAP processed imagery. With enhanced boundaries between landcover classes, such as the lake, subtle changes in these boundaries will be detectable and more accurately quantified. Finally, landcover classification and segmentation (possibly using unsupervised clustering for example), may yield more accurate and precise boundaries between large landcover classes.

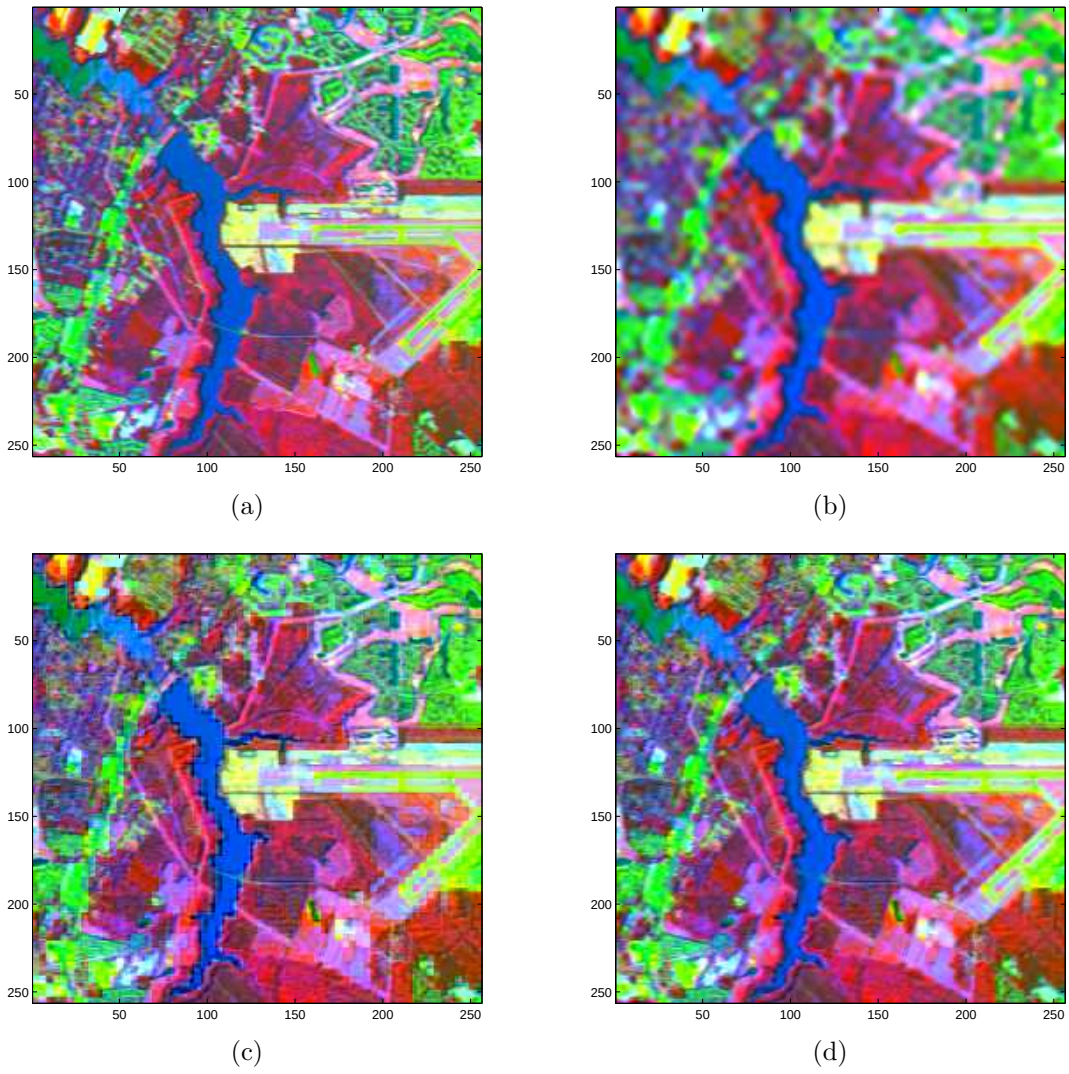


Figure 5.5: False color images showing the top three principal components for (a) the true high-resolution hyperspectral image (b) spline interpolated components (c) linear regression method (Price [2]) (d) the MAP estimate with $K = 16$.

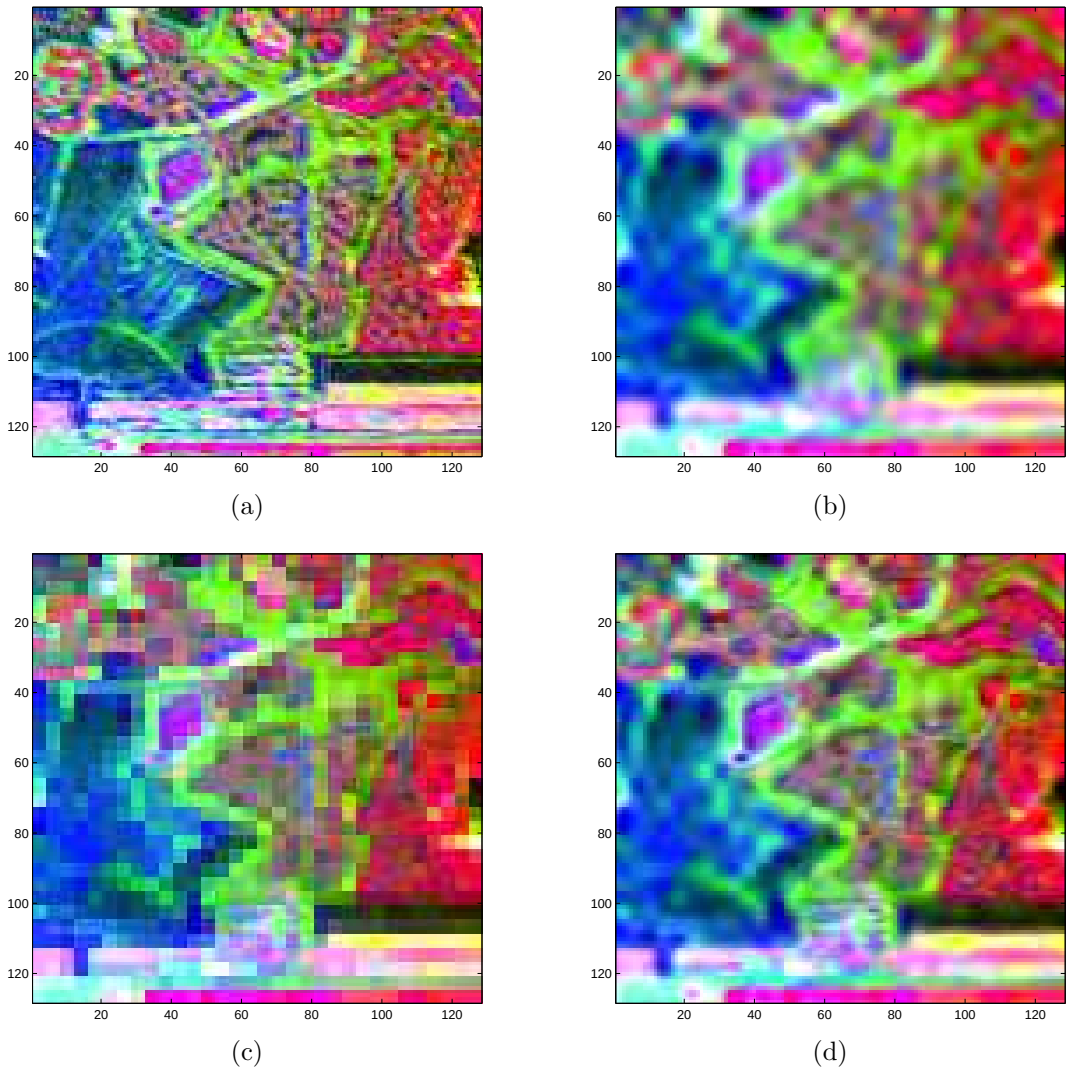
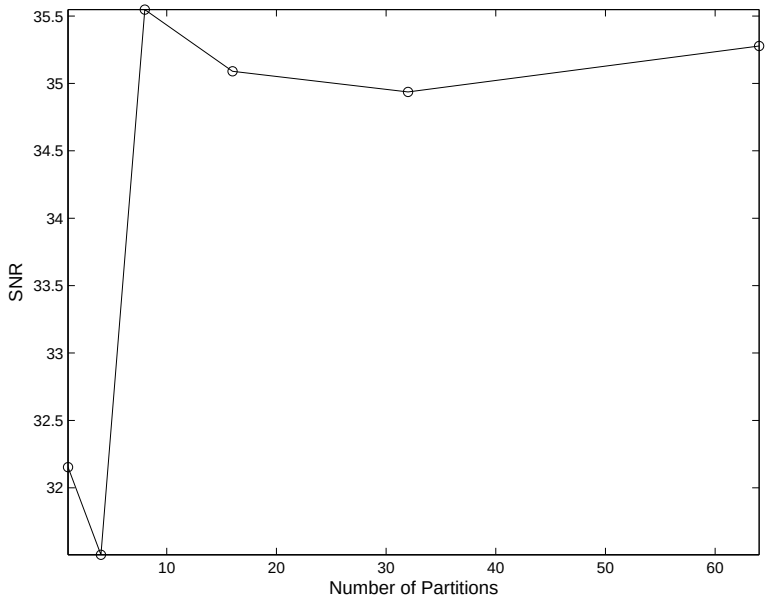
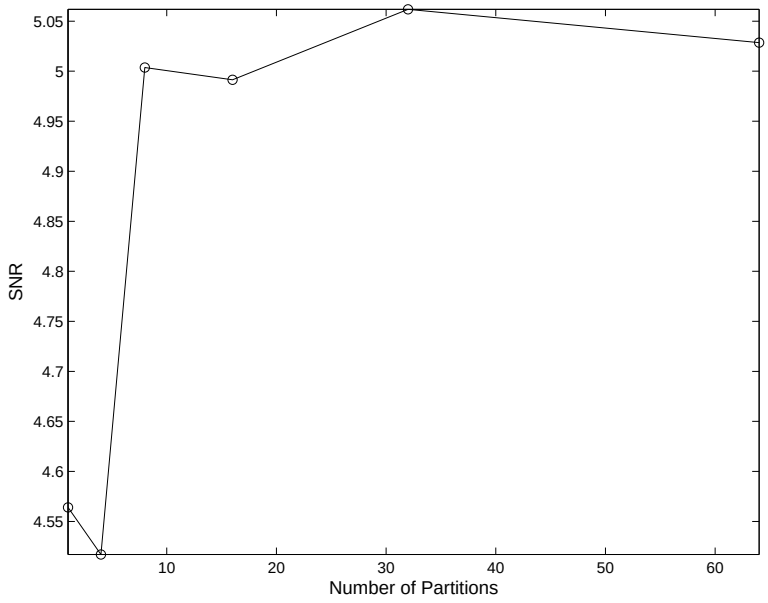


Figure 5.6: False color images showing principal components two, three, and four for (a) the true high-resolution hyperspectral image (b) spline interpolated components (c) linear regression method (Price [2]) (d) the MAP estimate with $K = 16$.



(a)



(b)

Figure 5.7: SNR versus the number of vector quantization partitions for (a) principal component one (b) principal component three.

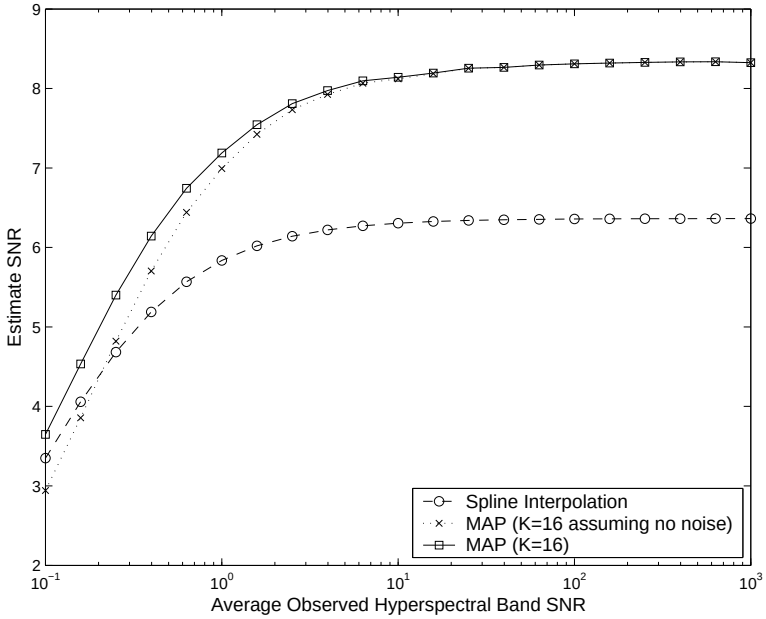


Figure 5.8: The SNRs of the estimates of principal component two as a function of the average SNR of the observed low-resolution hyperspectral bands.

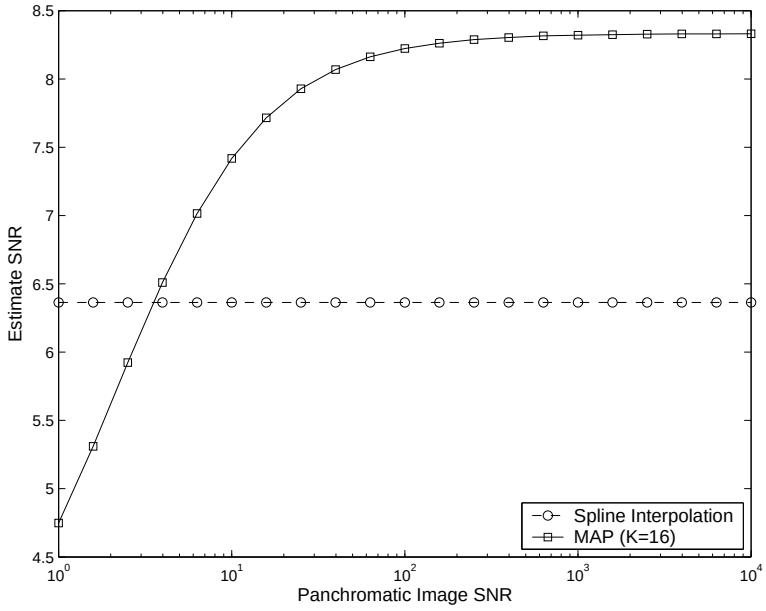


Figure 5.9: The SNRs of the estimates of principal component two as a function of the panchromatic image SNR.

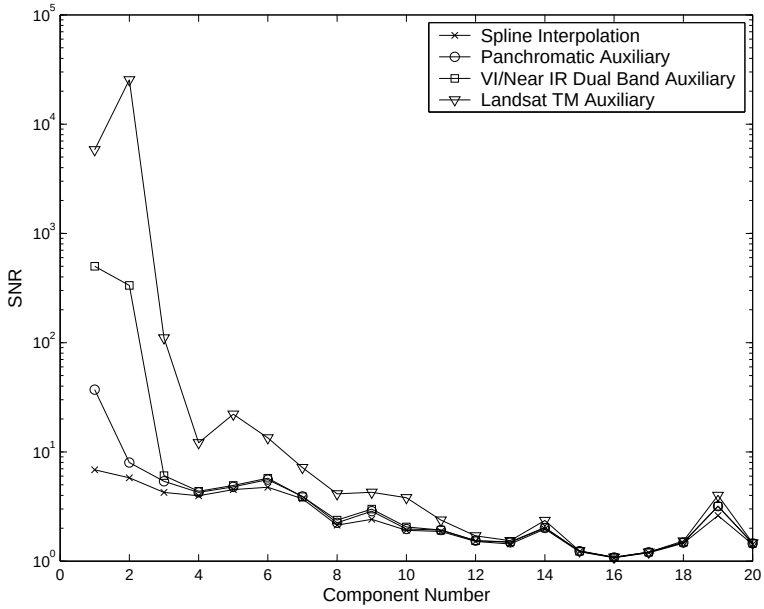


Figure 5.10: SNRs for spline interpolation and MAP estimator using simulated panchromatic, dual-band, and six-band landsat TM auxiliary senors.

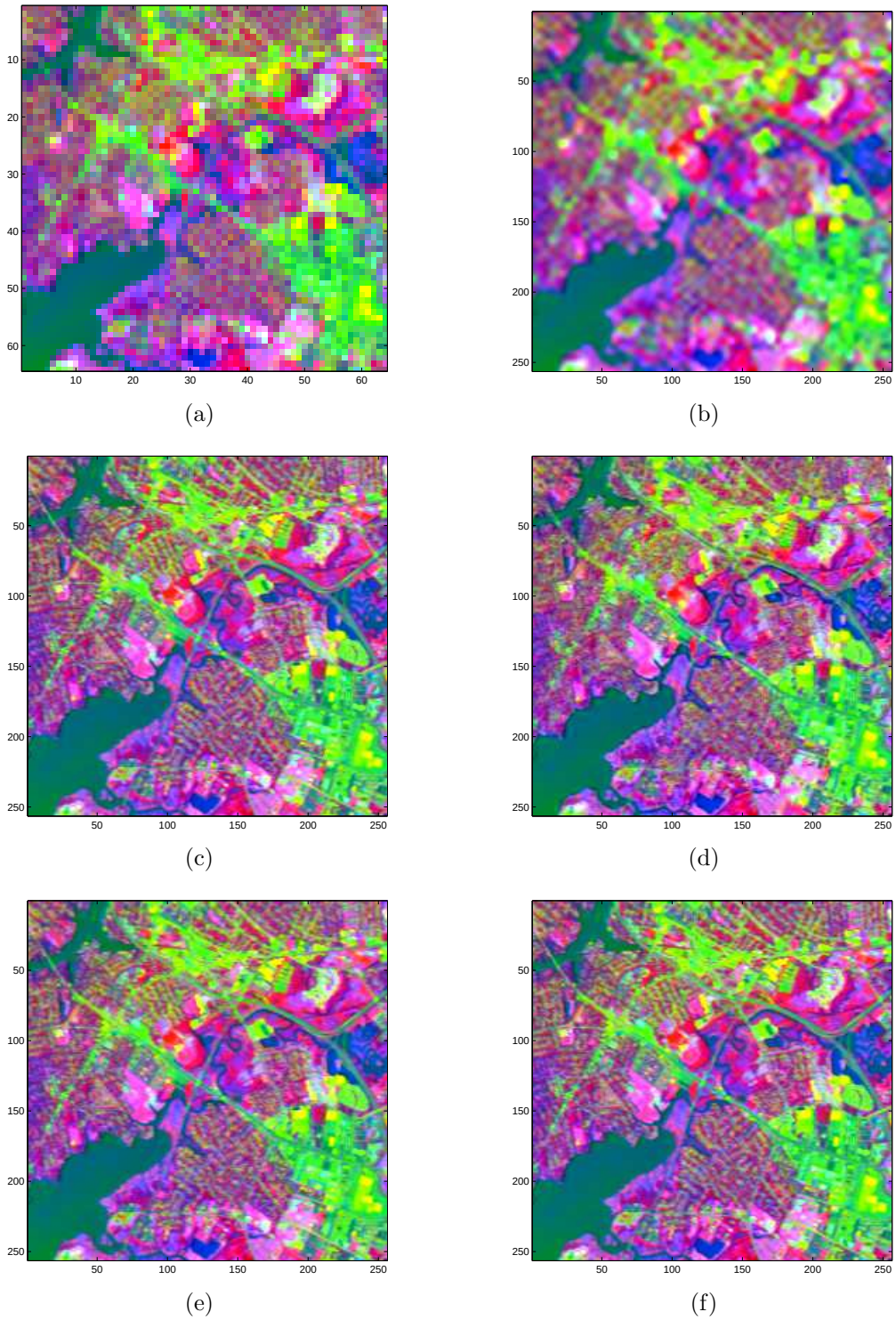


Figure 5.11: Principal components one, two, and three for (a) low resolution hyperspectral imagery (b) spline interpolation (c) true high resolution imagery (d) MAP estimate using a panchromatic auxiliary sensor ($K = 16$). (e) MAP estimate using a dual band auxiliary sensor (f) MAP estimate using a six-band auxiliary sensor.

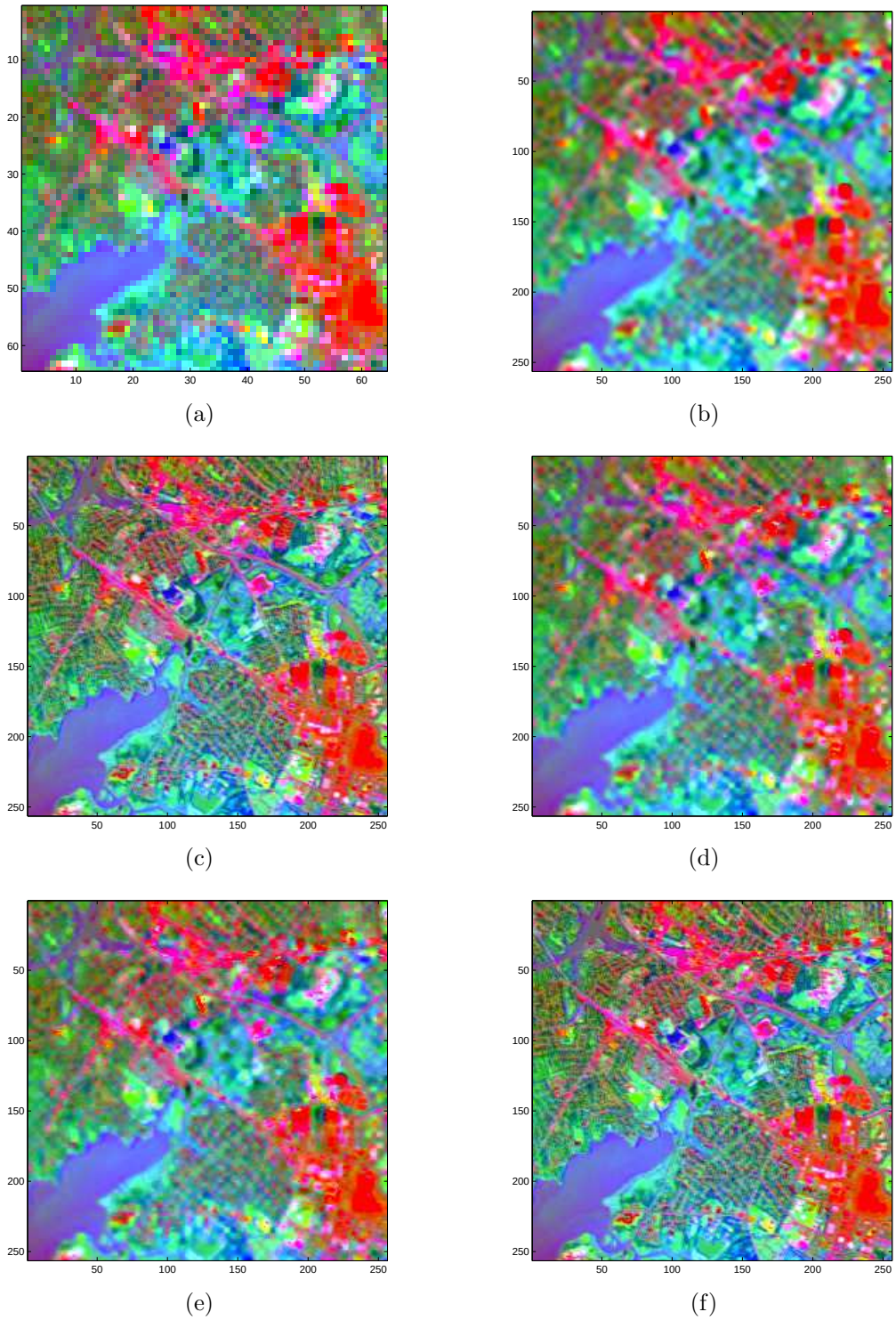


Figure 5.12: Principal components two, three, and four for (a) low resolution hyperspectral imagery (b) spline interpolation (c) true high resolution imagery (d) MAP estimate using a panchromatic auxiliary sensor (K=16). (e) MAP estimate using a dual band auxiliary sensor (f) MAP estimate using a six-band auxiliary sensor.

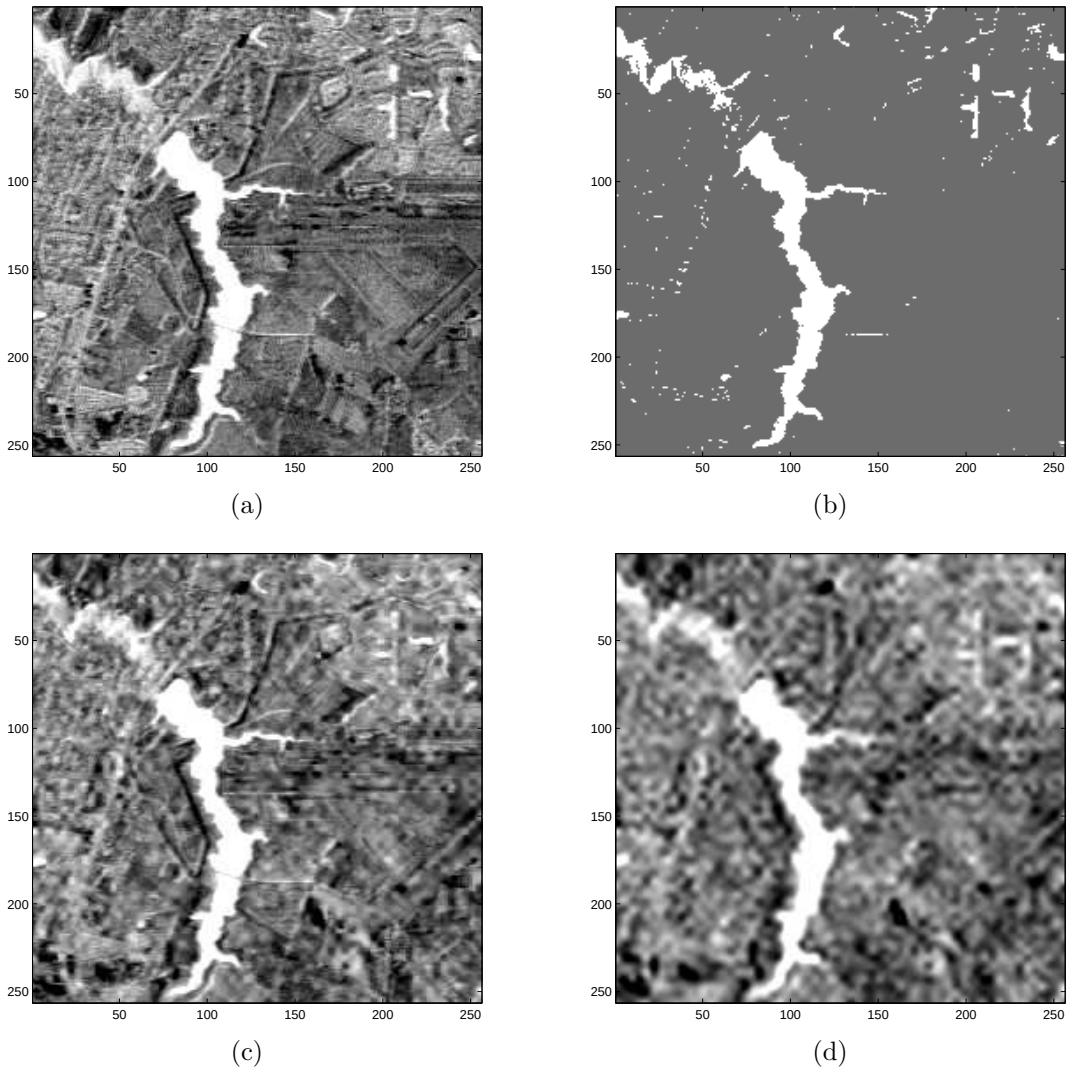


Figure 5.13: Matched filter results with target spectrum sampled from lake. (a) Matched filter output using the true high resolution imagery. (b) Binary detection map obtained by thresholding the matched filter output. Matched filter output using the (c) MAP estimator (K=16) and (d) spline interpolation.

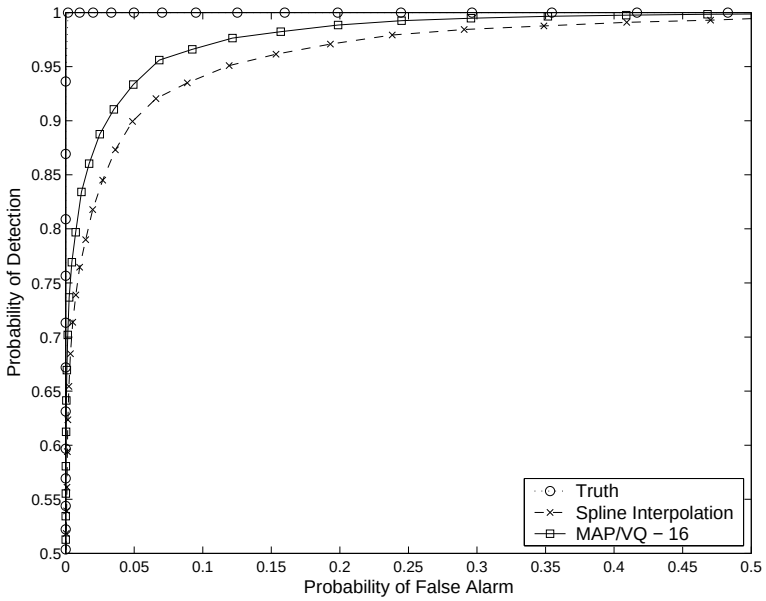


Figure 5.14: Receiver operating characteristic curve for the matched filter using the high resolution imagery (truth), MAP estimator, and spline interpolation.

Chapter 6

Mercury Multi Computer Implementation

Due to the assumptions made in Sections 4.3 and 4.11 regarding the PSF and covariance matrix structures, the algorithm for computing the MAP estimate $\hat{\boldsymbol{z}}$ may proceed one super-pixel at a time. On a serial machine, a loop over super-pixels is required which is the most CPU intensive portion of `MAP_Algorithm.m`.[1] Therefore, it is the loop over super-pixels which warrants a parallel implementation. Section 6.1 presents the speed improvements gained by use of the Mercury system as applied to the super-pixel loop. An important related topic is the use of single precision and its effects on timing and algorithm stability. This is discussed in Section 6.2 where diagonal loading is introduced to improve stability.

6.1 Timing Study

All timing results given in this section correspond to the most general (i.e. slowest) case where the $P \times P$ diagonal blocks of the conditional covariance matrix $C_{\tilde{z}|x}$ are different from one another (i.e. vary within super-pixels). Therefore, as a function of the number of PCA components chosen, all timings of the super-pixel loop represent the worst case. Due to having discovered a new way to speed up the super-pixel loop for the general case algorithm, these timing results are considerably better than what was earlier thought to be the case.

In our MAP implementation, one node was designated as the controller. The controller node receives input data from the host PC, and provides overall system control for the other nodes. After the other nodes receive partitioned data from the controller, all nodes then perform the desired processing in parallel.

[1] This assumes covariance inverses are required within the super-pixel loop, occurring for example when the linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is *not* assumed and VQ estimation of the conditional covariance is method 2.

We implemented our application using the Parallel Acceleration System (PAS), which is a C library provided by Mercury. Residing as an application layer over the standard Mercury operating system (MC/OS), PAS provides a convenient mechanism for writing code that scales to any number of nodes. It also allows for data to be sent from the controller to the processing nodes in either a BLOCKED or WHOLE manner, with usec synchronization between nodes.

All inputs necessary for computing the MAP estimate are written to a file by the host computer which is read by the controller node. The controller node then scatters the input to the worker nodes which (along with the controller) work on their share of super-pixels. Since the number of super-pixels is typically very large in comparison with the number of available nodes, super-pixels can be evenly spread among them. Since each super-pixel takes the same time to process as any other super-pixel, the work load will automatically be balanced.

Figures 6.1-6.4 show timing results as a function of available computational nodes. Since the curves shown were run on the AdapDev 1280, we were able to vary the number of available nodes from 0 to 8. The data plotted above 0 nodes is timing results for the host computer alone, which is an Intel machine running at 1266 MHz under Windows 2000. Data plotted above 1 node corresponds to running the entire super-pixel loop on one node (the controller node). Data plotted above 8 nodes corresponds to all 8 nodes computing the super-pixels MAP estimates in parallel.

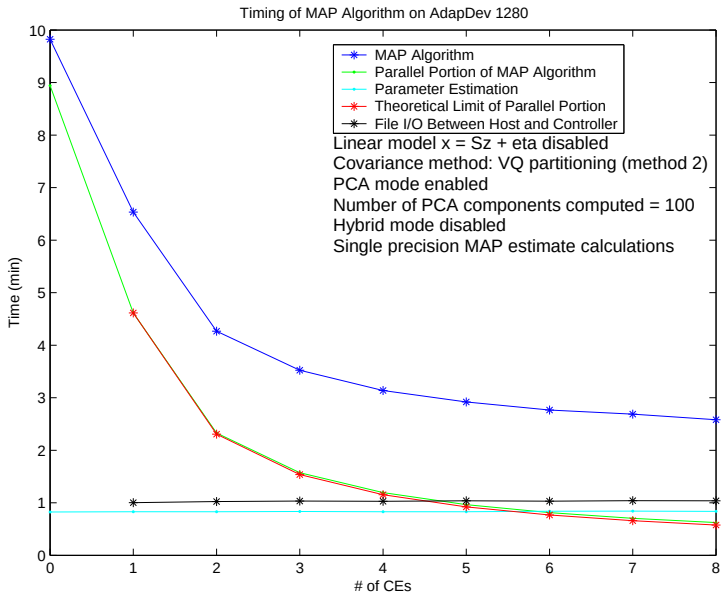


Figure 6.1: Timing of MAP Algorithm on AdapDev 1280 (100 PCA Components)

The blue curves represent the total time in minutes taken to run `MAP_Algorithm` as determined by the MATLAB `tic` and `toc` functions. The green curves represent the time taken to perform the CPU intensive loop over super-pixels, which is computed in parallel

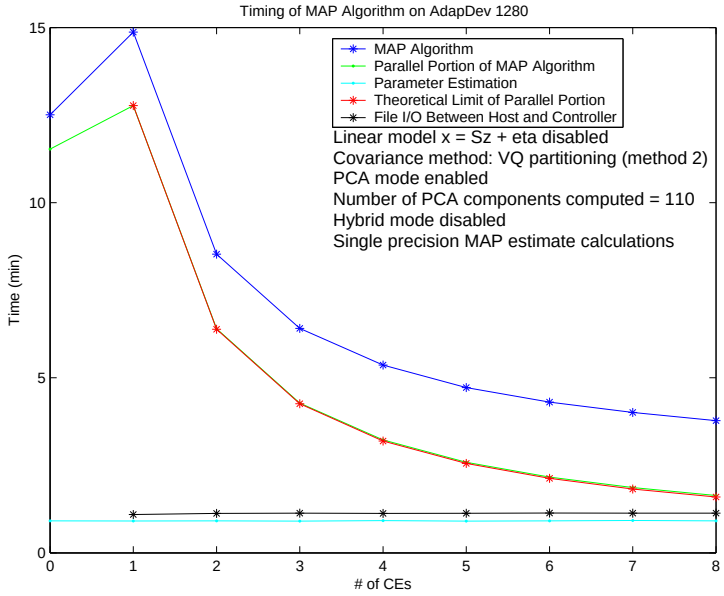


Figure 6.2: Timing of MAP Algorithm on AdapDev 1280 (110 PCA Components)

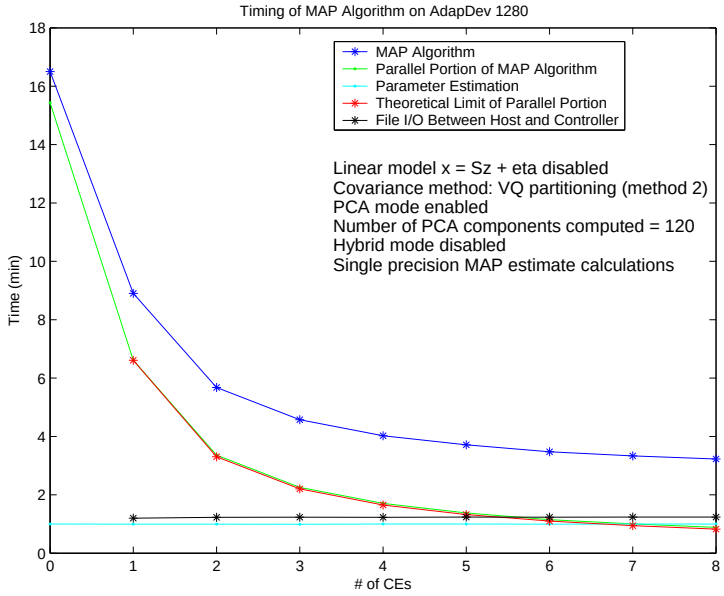


Figure 6.3: Timing of MAP Algorithm on AdapDev 1280 (120 PCA Components)

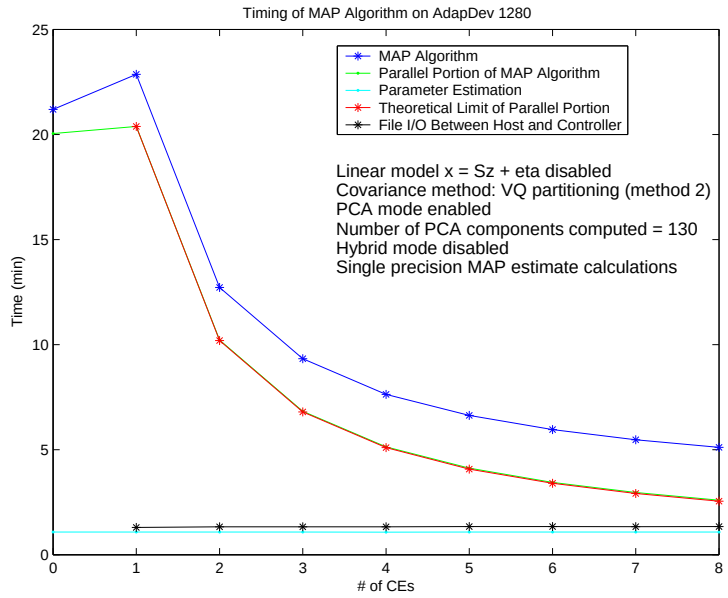


Figure 6.4: Timing of MAP Algorithm on AdapDev 1280 (130 PCA Components)

whenever the number of nodes is two or more. This does not include overhead time taken on the controller for file I/O from and to the host. However, it does include other small overhead times, such as communication between the controller and worker nodes, and the time to initialize nodes. The black curves represent total file I/O time between the host and controller nodes, and the cyan curve represents the time taken by the host to estimate parameters. Since parameter estimation is solely a MATLAB computation (i.e. not parallelized), and file I/O time is strictly between the host and the controller (i.e independent of the number of worker nodes), we see the cyan and black curves are nearly constant. The blue curve is well approximated as sum of the cyan, black, and green curves. That is, the total runtime of `MAP_Algorithm` consists mainly of three disjoint processes: parameter estimation, file I/O, and super-pixel computation.

The red curves represent a theoretical lower bound on the parallelized super-pixel loop, ignoring file I/O time on the controller from and to the host. The red curve is computed simply by dividing the green time for a single node, by the number of available nodes. Thus, the green and red data point for 1 node are identical. In each of Figures 6.1-6.4, the red curve is seen to follow the green curve quite closely, indicating that the work load is well balanced.

In Figures 6.1 and 6.3, we note that the time taken to run the super-pixel loop entirely on the host is significantly longer than the time taken to run it on a single compute node of the AdapDev 1280. This occurs even though the host runs at 1266 MHz while each Mercury CE runs at 500 MHz. The reason is the use of single precision and SAL function calls on the Mercury CE, which can dramatically speed up computations. However, in Figures 6.2 and 6.4, we see the reverse: the host runs faster than a single CE. Timing results on Mercury

compute nodes are very sensitive to data lengths coming into the various memory caches. Since the number of PCA components changes among the figures, so do the data lengths entering memory caches. Therefore, variation in runtime between the host and a single CE, is almost certainly due to changing data lengths entering memory caches.

The green curves of Figures 6.1-6.4 show exactly what we would expect if the work load is well balanced among the compute nodes. That is, the curves have a $1/N$ behavior just as the theoretical lower bound (red) curve has. Also, we see that the red and green curves are always close, which is further evidence of efficient load balancing. We may also ascertain that work loads are well balanced by examining Figure 6.5, where 150 PCA components have been processed on 8 nodes. In that figure, we see a time line of events for each computation node. In particular, the green segments of the plot represent the processing time of a node, i.e. the time it took to process all of its assigned super-pixels, not counting any communication time or idle time. The start of a green bar represents the time at which a node received all data necessary to compute its super-pixels, and the end of one represents the time at which the node finished its last super-pixel. Since the green bars across each node have essentially the same length (about 246 seconds), we see that the work loads are well balanced.

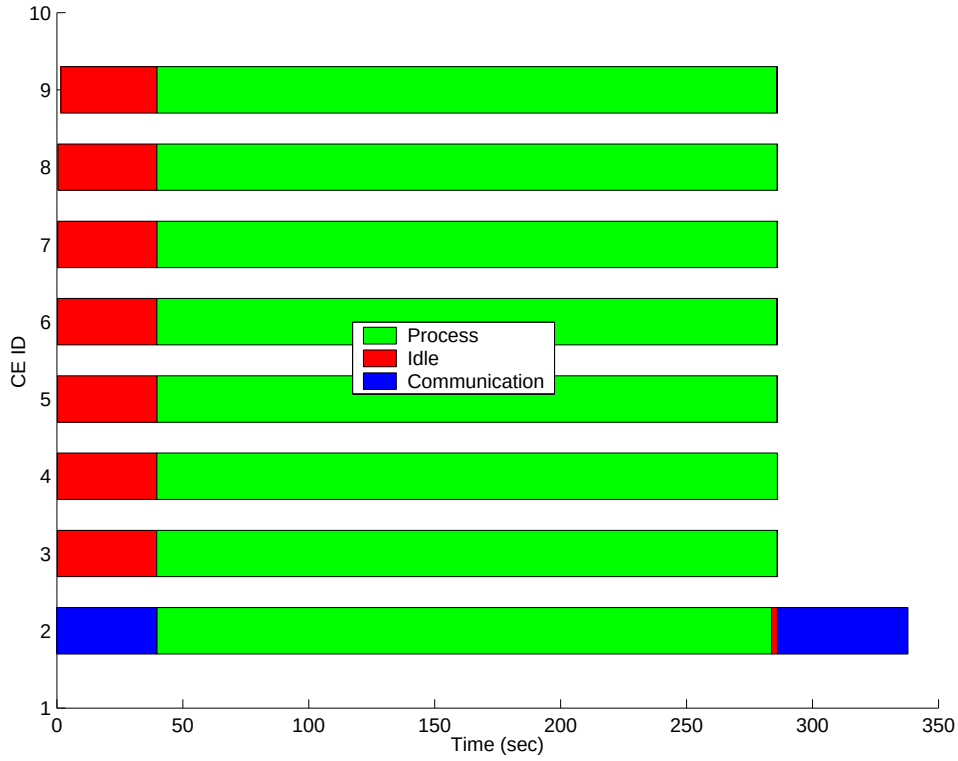


Figure 6.5: Time Line for 150 PCA components on the AdapDev 1280

Also seen in Figure 6.5 are blue bars representing communication time and red bars representing idle wait time. The controller (i.e. CE2) spends about 40 seconds during its initial communication phase. This consists mostly of reading data from the host computer

but also includes a small amount of time pushing data to the local memory of the worker nodes. Each of the 7 worker nodes (i.e. CE3-CE9) spends about the same amount of time waiting for the controller node to finish its initial communication phase. After a worker finishes processing its super-pixels, it has a short wait while the controller pulls data from its local memory. Since the controller was the first to finish its processing of super-pixels, it also has a short wait while the worker nodes to complete their processing. The final communication time of the controller consists of a negligible time pulling data from the workers, followed by a much longer time writing a file used by the host computer. No communication time is seen at any worker node because data is *pulled* by the controller from the local memory of each worker, as opposed to *pushed* to the controller by the workers.

In order to generate Figure 6.5, a series of time stamps are inserted into C code for running on the Mercury. These time stamps are identical to those for use with Mercury's TATLView post processing program. After running the code, a series of event times are written to an ASCII file. Although this file can be read by the TATLView program to view the resulting time line, we have instead chosen to read and process it with MATLAB. The result is a plot such as that given by Figure 6.5.

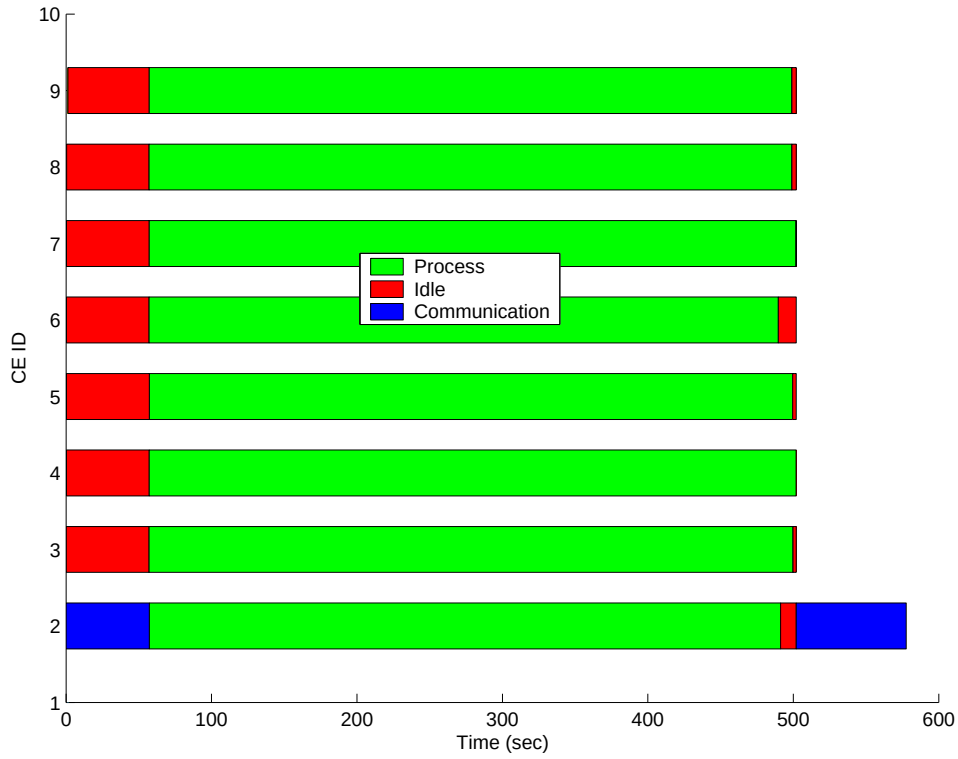


Figure 6.6: Time Line for 224 PCA components run on the AdapDev 1280

Figure 6.6 shows another example running the AdapDev 1280 with 8 nodes. By taking the number of PCA components to be 224, we have increased the size of the matrices to invert to 224. Qualitatively, the green, red, and blue bars show the same behavior as Figure 6.5. Quantitatively, we see that the super-pixel loop took approximately 445 seconds

and file I/O on the controller took about 132 seconds. The entire run of `MAP_Algorithm` including parameter estimation and file I/O on the host, took about 11.7 minutes. This case constitutes the full AVIRIS data set, and is the largest problem we have run to date.

Figure 6.7 shows the same example ($P = 224$, $N = 256^2$, and $M = 64^2$), run on the 64 node Mercury (Hawk) located at WPAFB. Each node of Hawk is a G4 Power PC running at 400 MHz and has 128MB of memory per node. This run was important in order to verify the portability (and scalability) of the code, and to verify that memory limitations on Hawk are not exceeded. Memory is a concern because the input data for all super-pixels resides in the local memory of the controller prior to scattering it to the worker nodes. The fact that the program ran, is conformation that we did not exceed the 128MB upper limit.

It is seen in Figure 6.7 that each of 64 nodes took approximately 63 seconds to complete the parallelized super-pixel loop. From Figure 6.6, we see it took each of 8 nodes 445 seconds for the same computation. Since each machine processed 64^2 super-pixels, the AdapDev 1280 processed approximately 1.15 super-pixels per second per node, while Hawk processed approximately 1.02 super-pixels per second per node. The different rates are explained by the different processing speeds of the nodes: 500 MHz for the AdapDev 1280 and 400 MHz for Hawk. We also see that file I/O between the host of Hawk and the controller CE is considerably slower than that of the AdapDev 1280. The host of Hawk runs under a Solaris operating system with limited memory and slow disk access. While Hawk might be useful for testing a real time implementation of a MAP algorithm based on Equation (4.70), further use of Hawk for the existing implementation is unwarranted.

6.2 Effects of Single Precision and Diagonal Loading

Our approach to developing code to run on a Mercury system consists of five basic steps. First develop a working code in MATLAB, second write a C-version to run under Windows, third modify the Windows C code to make (emulated) SAL function calls, fourth port this code to the Mercury for running on a single node, and fifth use PAS library calls to parallelize it for use on several nodes. Each step in the process required checking the answers obtained with the original MATLAB version.

Of the four Mercury systems we tested, the compute nodes have been G4 Power PC 7400 microprocessor (or compatible) with AltiVec technology. In order to take better advantage of this architecture, running in single precision is important. Our experiments indicate that a 5 fold speed increase can be expected in changing from single precision to double precision. There are several reasons for this:

- Use of the vector processing unit (VPU) is restricted to single precision.
- AltiVec language extensions are restricted to single precision.
- The floating point unit is twice as fast for single precision.

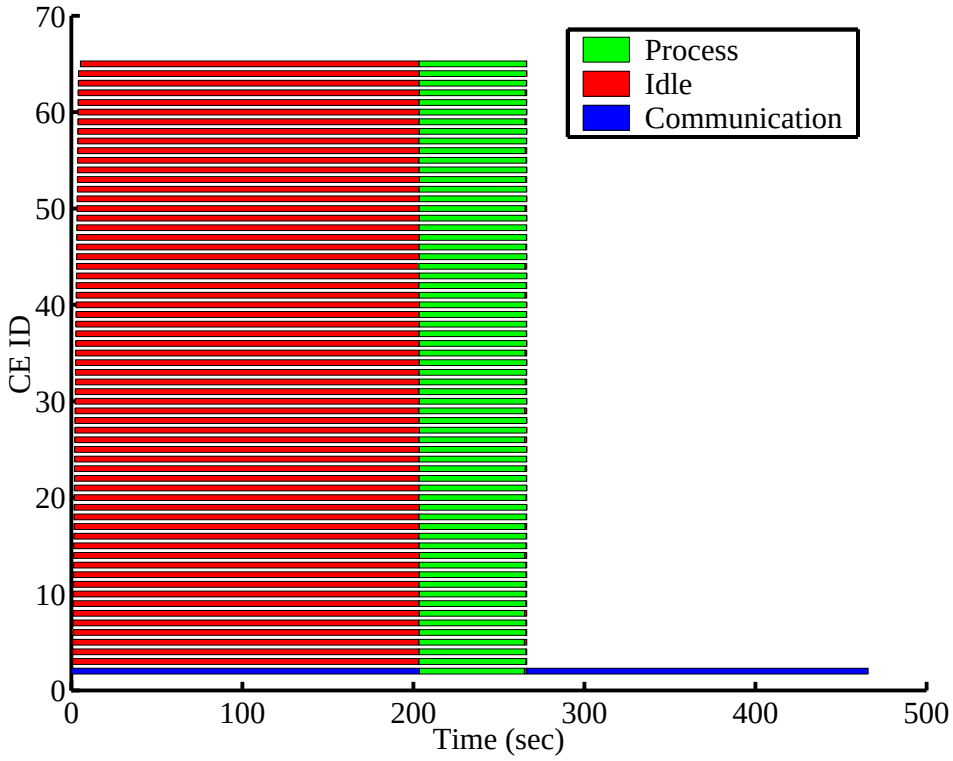


Figure 6.7: Time Line for 224 PCA components run on Hawk using 64 Nodes

- Memory caches are twice as efficient (each element takes half as much space).
- The RACEway is twice as efficient (half as many bits per word).

Fortunately, the use of double precision is not necessary for the portion of the code that was parallelized. This is true even in light of the fact that poorly conditioned matrices may arise whose inverses are required. If nothing is done to improve the conditioning of these matrices, the inverse computation becomes unstable resulting in MAP estimates that do not match double precision MATLAB code. However, the matrices to be inverted are diagonally loaded when a positive value of σ_n^2 (input variable `var_y` of `MAP_Algorithm.m`) is supplied, resulting in improved conditioning. The single precision C code running in parallel on a Mercury and the MATLAB version on the host PC yield nearly identical results in all cases tested provided an appropriate diagonal load σ_n^2 is applied.

Applying a diagonal load does raise some issues. One issue is that supplying a diagonal load may conceivably have a negative effect on SNR. This fortunately does not occur. In fact, supplying a diagonal load has a *positive* effect on average SNR as indicated in Figures 6.8-6.11, where all computations were performed in double precision MATLAB. Another issue is how to set the diagonal load. From the same figures we see that there are a wide range of values we may use for the diagonal load, all of which improve average SNR.

As shown in Figure 6.8 for example, we see that any diagonal load greater than 0 and less

than or equal to 30 results in an improved average SNR. The optimal load as determined by the maximum average SNR is somewhere near 12. Although the optimal load is not readily determined a priori, our experimentation indicates that just about any number between say 1 and 50 should improve average SNR (compared to no loading) while providing acceptable conditioning for single precision matrix inversion.

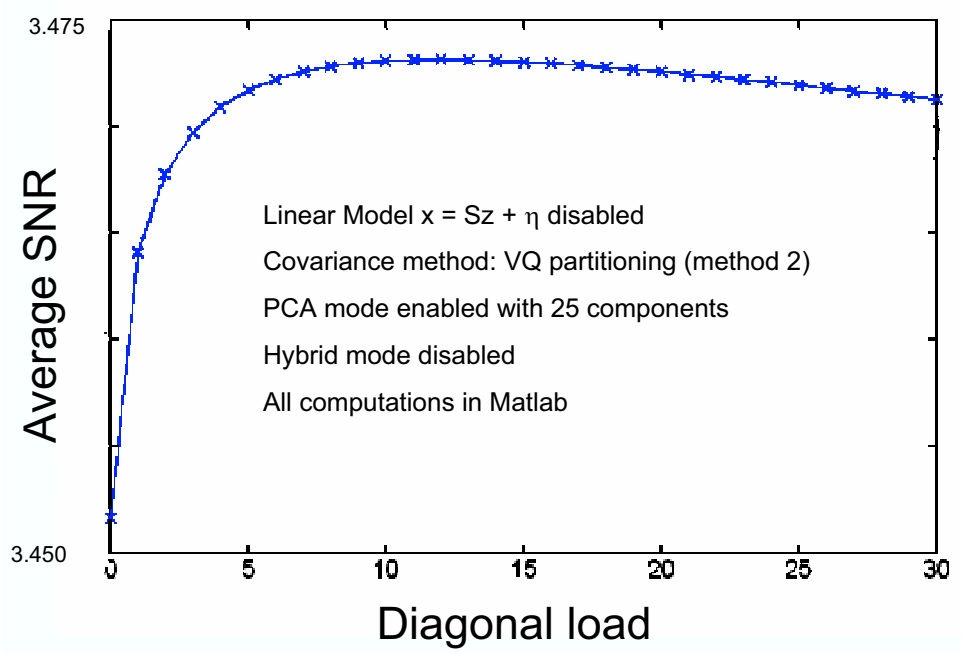


Figure 6.8: Effect of Diagonal Loading on SNR (25 PCA Components)

An additional advantage to using single precision over double precision is the size of the problem that can be loaded onto the Mercury system. Since a single precision variable takes half the memory of a double, the matrices involved can be twice as big. This is particularly important because the local memory on a node is limited. On the AdapDev 1280 and the 96 node Mercury at WPAFB (Titan), it is 256MB per node, and on the 64 node system at WPAFB (Hawk) it is 128MB per node. Since the controller must hold data for all super-pixels, the size of the problem is limited to the available memory of one node.

In summary, the use of single precision is desirable on two fronts: speed and memory. Although its use requires diagonally loading, that requirement is desirable on its own merits as measured by improved average SNR. The only drawback to diagonally loading is the additional user input of specifying the load. However, our experimentation indicates that there is a wide range of acceptable loads, any of which will improve average SNR while simultaneously conditioning the matrices involved for a stable inverse in single precision.

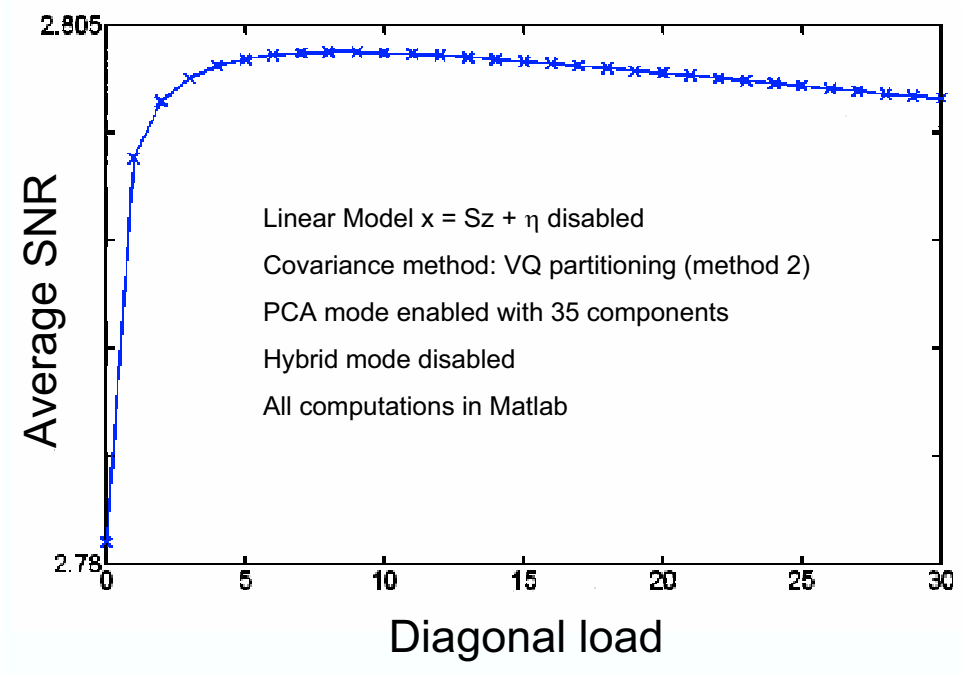


Figure 6.9: Effect of Diagonal Loading on SNR (35 PCA Components)

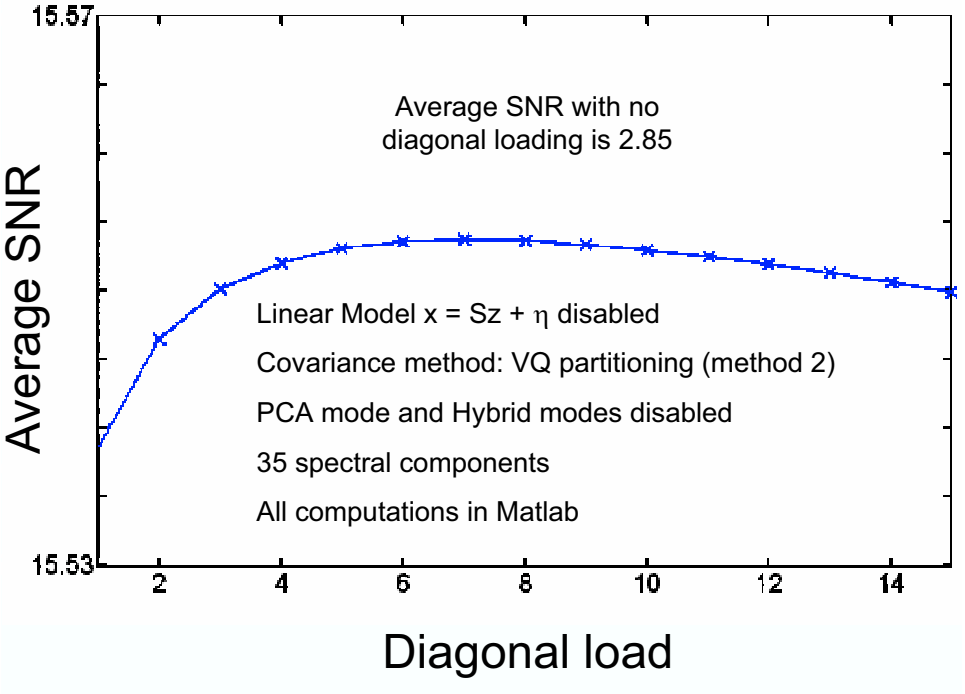


Figure 6.10: Effect of Diagonal Loading on SNR (35 Spectral Components)

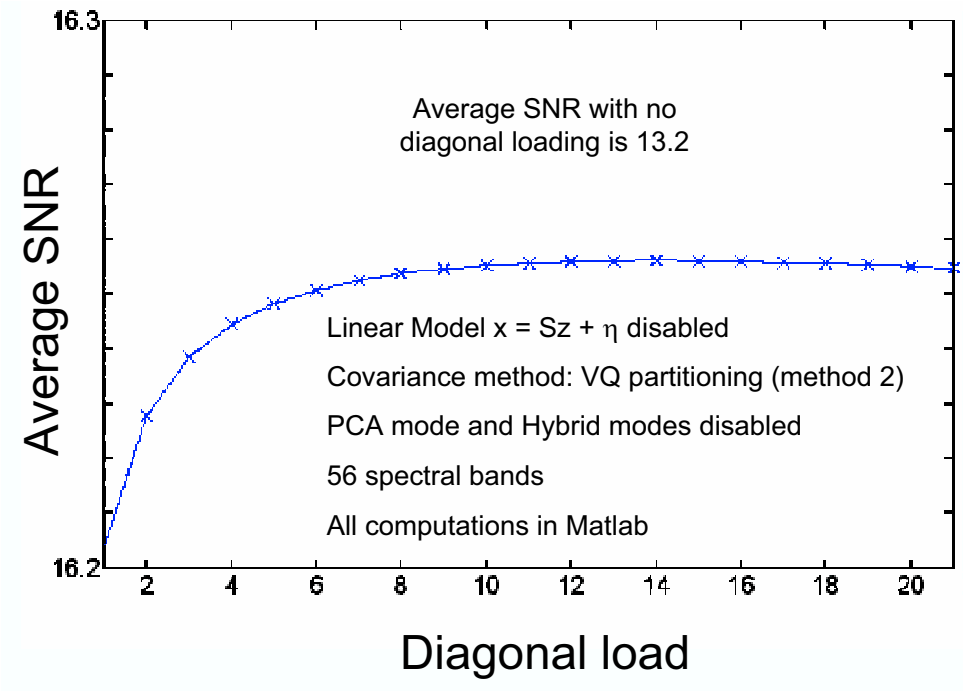


Figure 6.11: Effect of Diagonal Loading on SNR (56 Spectral Components)

Chapter 7

Stochastic Mixing Model: Interface to the MAP Algorithm

Two separate interfaces are presented, one where the linear model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is assumed and one where it is not. In the former case, we provide the expressions necessary for estimating the required unconditional means and covariances and in the later for the required conditional means and covariances. In either case, high resolution abundance maps, end-member mean vectors, end-member covariances, and a PCA transformation matrix are required as input. Only the case where the linear model is assumed (discussed first) has been incorporated into `MAP_Algorithm.m`.

7.1 SMM Interface Assuming a Linear Model

The stochastic mixing model [25–28], is similar to the well-developed linear mixing model in that it attempts to decompose spectral data in terms of a linear combination of endmember spectra. However, in the case of the stochastic mixing model, the endmembers $\boldsymbol{\varepsilon}_k$ are $P \times 1$ normally-distributed random vectors, parameterized by their mean vector $\mathbf{m}_{\varepsilon_k}$ and covariance matrix $\boldsymbol{C}_{\varepsilon_k}$, instead of deterministic spectra. The endmember index k ranges from one to N_e, the number of endmembers assumed in the scene. A finite number, N_c, of mixture classes are defined as linear combinations of the endmember random vectors. These mixture class random vectors are defined as

$$\boldsymbol{\omega}_q \;=\; \sum_{k=1}^{N_e} a_k(q)\boldsymbol{\varepsilon}_k, \tag{7.1}$$

where $a_k(q)$ are the endmember abundances associated with the mixture class, and $q = 1, 2, \ldots, N_c$ is the mixture class index. The endmember abundances comprising the mixture classes are assumed to conform to the physical constraints of being between zero and one, and summing to unity, and are quantized at a specified level. This is done by a structured,

iterative search algorithm that determines all the possible combinations of quantized abundances (i.e., discrete levels between zero and one) for the specified number of endmembers that satisfy the physical constraints. The result is a finite and discrete set of mixture classes.

Each low resolution hyper-pixel $\vec{y}_m$ of the observed hyperspectral image is then assumed to be a realization of a particular mixture class random vector. Thus, after the SMM algorithm converges, $\vec{y}_m$ is assigned to a corresponding mixture classe q_m, for $m = 1, 2, \ldots, M$. From this low resolution mixture class map, low resolution abundance maps, $a_k(q_m)$ are formed. That is, there are N_e abundance maps giving the spatial abundance distributions for each endmember. Treating $\vec{y}_m$ as a random vector, we have

$$\vec{y}_m = \sum_{k=1}^{N_e} a_k(q_m)\boldsymbol{\varepsilon}_k, \tag{7.2}$$

The endmember statistics are estimated from a subset of the observed spectra. Specifically, the sample mean and covariance estimated are computed for each endmember from the observations $\vec{y}_m$ for $m \in \Omega_k$, where Ω_k is the set of spatial positions assigned to endmember class k in the SMM algorithm. For more detail on the estimation processes involved in formulating an SMM, see [25–28].

To get a spatially-varying statistical model at the high resolution, we begin by bilinearly interpolating the low resolution abundance maps to give the high resolution abundance maps $a_{i,k}$, for $i = 1, 2, \ldots, N$, and $k = 1, 2, \ldots, N_e$. Next we assume that the high resolution hyper-pixels obey the linear mixing relationship using the endmembers estimated from the low-resolution data. Treating the high resolution hyper-pixel $\vec{z}_i$ as a random vector, we have:

$$\vec{z}_i = \sum_{k=1}^{N_e} a_{i,k}\boldsymbol{\varepsilon}_k, \tag{7.3}$$

Thus, the spatially-varying mean and covariance for the high resolution hyper-pixels are given by:

$$\boldsymbol{m}_{z_i} = \sum_{k=1}^{N_e} a_{i,k}\boldsymbol{m}_{\varepsilon_k}, \tag{7.4}$$

$$\boldsymbol{C}_{z_i} = \sum_{k=1}^{N_e} a_{i,k}^2\boldsymbol{C}_{\varepsilon_k}. \tag{7.5}$$

To incorporate the SMM into the MAP estimator when the model $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is assumed, the means and covariances of the pdf for $\vec{z}_i$ are given by (7.4) and (7.5). Thus, what would be needed from the SMM is the endmember means and covariances and the high-resolution abundance maps (or the low-resolution abundance maps, that we could bilinearly interpolate). The endmember mean vectors and covariance matrices could be provided in

the spectral space or the PCA space (or simply the leading PCA subspace). If provided in spectral space, and processing is to be done the PCA space, they can be transformed (using the PCA transformation matrix).

If the endmember means and covariances and abundance maps are provided in a leading PCA subspace, we need the PCA transformation matrix used as well. This way we are sure to transform the imagery to be processed in a manner consistent with that used to compute the SMM parameters. This is due to possible variation in how the Eigen vectors of the spectral data are computed in performing the PCA transformation. Estimates of the lower PCA subspace can be done using interpolation.

7.2 SMM Interface Without Assuming a Linear Model

Incorporating the SMM into the MAP estimator when $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$ is *not* assumed requires a somewhat more complex process. In this case, consider joint endmember random vectors containing the hyperspectral endmember and the corresponding spectra for the auxiliary image. In particular, let the joint endmembers be $(P + \nu) \times 1$ random vectors

$$\boldsymbol{u}_k = \left[\boldsymbol{\varepsilon}_k^T, \boldsymbol{\alpha}_k^T\right]^T, \tag{7.6}$$

where $\boldsymbol{\alpha}_k$ is a $\nu \times 1$ random endmember vector associated with the auxiliary sensor. The joint endmember random vector has mean $\boldsymbol{m}_{u_k}$ and covariance $\boldsymbol{C}_{u_k}$. The joint endmember statistics are estimated using the sample mean and covariance from the observations $\bar{\boldsymbol{w}}_m = [\bar{y}_m^T, \bar{\boldsymbol{x}}_m^T]^T$ for $m \in \Omega_k$, where the $\bar{\boldsymbol{x}}_m$ are artificially degraded spectra from the auxiliary sensor. That is, the high resolution multispectral sensor data are degraded to a spatial resolution to match the observed hyperspectral data. Now define a joint random vector at the high resolution as follows

$$\boldsymbol{w}_i = \left[\bar{z}_i^T, \boldsymbol{x}_i^T\right]^T. \tag{7.7}$$

We will model this as a linear combination of the joint endmember random vectors. As before, the relative abundances will be given from the high resolution abundance maps, $a_{i,k}$, for $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, N_e$. Thus,

$$\boldsymbol{w}_i = \sum_{k=1}^{N_e} a_{i,k}\boldsymbol{u}_k. \tag{7.8}$$

Note that we are using joint endmember statistics estimated at the lower resolution for the higher resolution estimation problem. We are also relying on bilinear interpolation to generate high resolution abundance maps. Given the linear relationship in (7.8), the spatially-varying mean and covariance for $\boldsymbol{w}_i$ is related to the joint endmember class statis-

tics by

$$\boldsymbol{m}_{w_i} = \sum_{k=1}^{N_e} a_{i,k}\boldsymbol{m}_{u_k}, \tag{7.9}$$

$$\boldsymbol{C}_{w_i} = \sum_{k=1}^{N_e} a_{i,k}^2\boldsymbol{C}_{u_k}, \tag{7.10}$$

where $i = 1, 2, \ldots, N$. Because $\boldsymbol{w}_i$ is a joint random vector, the means and covariances will be of the form

$$\boldsymbol{m}_{w_i} = \left[\boldsymbol{m}_{z_i}^T, \boldsymbol{m}_{x_i}^T\right]^T, \tag{7.11}$$

$$\boldsymbol{C}_{w_i} = \begin{bmatrix} \boldsymbol{C}_{z_i,z_i} & \boldsymbol{C}_{x_i,z_i} \\ \boldsymbol{C}_{z_i,x_i} & \boldsymbol{C}_{x_i,x_i} \end{bmatrix}. \tag{7.12}$$

The conditional mean vector and covariance matrix for each spatial position are related to the joint statistics extracted from the right-hand-side of (7.11) and (7.12) using

$$\boldsymbol{m}_{z_i|x_i} = \boldsymbol{m}_{z_i} + \boldsymbol{C}_{z_i,x_i}\boldsymbol{C}_{x_i,x_i}^{-1}\left[\boldsymbol{x}_i - \boldsymbol{m}_{x_i}\right] \tag{7.13}$$

$$\boldsymbol{C}_{z_i|x_i} = \boldsymbol{C}_{z_i,z_i} - \boldsymbol{C}_{z_i,x_i}\boldsymbol{C}_{x_i,x_i}^{-1}\boldsymbol{C}_{x_i,z_i} \tag{7.14}$$

Thus, to incorporate the SMM statistics into the MAP estimator when the linear model relating $\boldsymbol{x}$ and $\boldsymbol{z}$ is not assumed, we require the joint endmember statistics $\boldsymbol{m}_{u_k}$ and $\boldsymbol{C}_{u_k}$. We would also require the high resolution abundance maps. With this information we would employ (7.9) and (7.10). Next we would use the right hand side of (7.11) and (7.12) to complete (7.13) and (7.14). Again, if the information is provided in a leading PCA subspace, we would require the PCA transformation matrix used to ensure consistency. Interpolation would be used for the lower PCA subspace.

Chapter 8

Misregistration Effects

During the course of hyperspectral image collection, the high-resolution broadband image ($\boldsymbol{x}$) and low-resolution hyperspectral image ($\boldsymbol{y}$) can become misregistered, which means that the spatial alignment of the two images is no longer consistent. Misregistration can occur for reasons such as a slight sensor misalignment or noise from various origins. One of the primary assumptions of the enhancement algorithm developed under this program effort is that the two images are in perfect alignment. The purpose, then, of the examining the performance of the algorithm under less-than-ideal misregistration/misalignment conditions is discover the algorithm's robustness to this effect, given that there exists a real chance of image misregistration in fielded hyperspectral sensor systems.

There are many types of misregistration that can occur such as translation, rotation, stretching, and shear, or any combination thereof. For these misalignment types, the misregistration errors that occur in-plane are by far the easiest to compensate for, because an inverse transformation exists if the mapping is one-to-one. Translation and rotation are in-plane transformations for which a simple inverse transformation usually exists. This inverse transformation can easily be applied to undo the misregistration, and consequently, bring the images back into alignment. Conversely, out-of-plane transformations are very difficult to deal with given that the mapping is usually one-to-many/many-to-one, and finding the inverse transformation can be impossible.

The types of misregistration explored here are restricted to translation and rotation (in-plane transformations), although in fielded systems any type is possible. The rationale for restricting the misregistration types is predicated on the fact that if the algorithm is sensitive to even these types of errors, it will definitely be sensitive to other types such as shear or stretching given these errors are generally more severe.

For the experiments presented, the enhancement quality was measured on a pixel-by-pixel basis between the original high-resolution hyperspectral image ($\boldsymbol{z}$) and MAP enhanced hyperspectral image ($\hat{\boldsymbol{z}}$). The error metrics were the mean squared error (MSE), the mean

absolute error (MAE), and the signal-to-noise ratio (SNR) given by Equation 8.1.

$$\begin{aligned} \text{MSE} &= \tfrac{1}{N}\sum_{i=1}^{P}\sum_{n=1}^{N}(z_{i,n}-\hat{z}_{i,n})^2, \\ \text{MAE} &= \tfrac{1}{N}\sum_{i=1}^{P}\sum_{n=1}^{N}|z_{i,n}-\hat{z}_{i,n}|, \\ \text{SNR} &= \textstyle\sum_{i=1}^{P}\sum_{n=1}^{N}(z_{i,n}-\mu_i)^2 / \sum_{i=1}^{P}\sum_{n=1}^{N}(z_{i,n}-\hat{z}_{i,n}-\mu_i^*)^2, \end{aligned} \tag{8.1}$$

where $\mu_i = \frac{1}{N}\sum_{n=1}^{N} z_{i,n}$ and $\mu_i^* = \frac{1}{N}\sum_{n=1}^{N}(z_{i,n}-\hat{z}_{i,n})$.

The data utilized was a single AVIRIS image, where the broadband image was created by averaging the hyperspectral bands of the original image. The low-resolution hyperspectral image was created by applying a 4x4 averaging PSF to the original hyperspectral image. For comparison, the enhancement algorithm was evaluated against a random broadband image as well as spline interpolation of the low-resolution hyperspectral image. The enhancement algorithm was run in PCA mode, keeping $P = 40$ principal components.

8.1 Rotation Experiments

The first set of experiments conducted involved rotating the high-resolution broadband image a small amount about the center of the image to bring the imagery out of alignment. The rotation transformation is defined by Equation 8.2

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}, \tag{8.2}$$

where θ is the angle of rotation, $[i\ \ j]^T$ are the original coordinates, and $[i'\ \ j']^T$ are the coordinates of the rotated broadband image. After the image was rotated, the image was cropped back to the original image dimensions and imputed in the enhancement algorithm. Figure 8.1 shows the enhancement results as a function of rotation angle (θ).

Figure 8.1 clearly demonstrates that even a small misregistration error can reduce performance significantly. After about 1^o of rotation, the map algorithm performs at the level of the baseline techniques.

8.2 Translation Experiments

The next experiments were conducted for the case of translation (both vertical and horizontal simultaneously) of the broadband panchromatic image relative to the low resolution

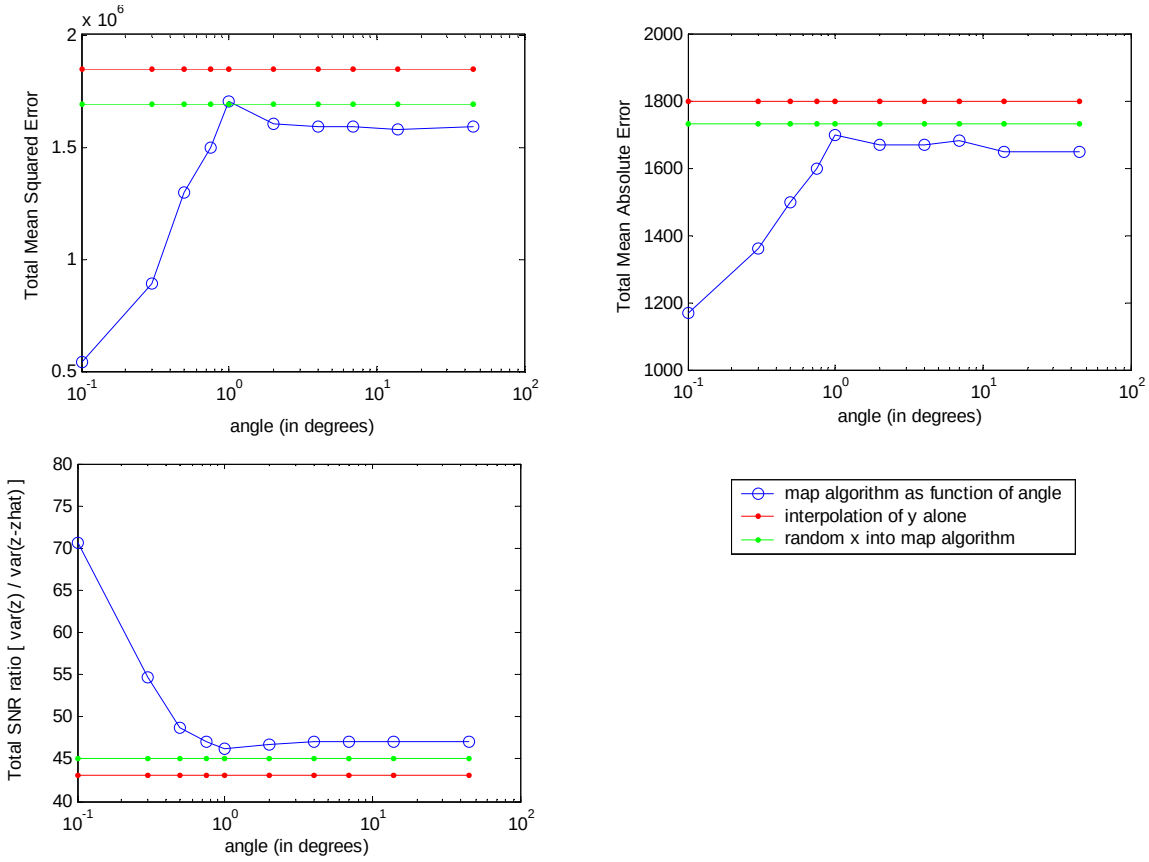


Figure 8.1: Rotation misregistration

hyperspectral image. The simple equation for this transformation is given by:

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} a \\ a \end{bmatrix} + \begin{bmatrix} i \\ j \end{bmatrix}, \tag{8.3}$$

where a is translation offset factor. The results are summarized in the Figure 8.2.

These results reflect the rotation results showing that once the broadband image is perturbed by even 1 pixel, the map algorithm is no longer effective. Even a misregistration error of a fraction of a pixel can degrade performance substantially.

8.3 Misregistration Conclusion

Overall then, these results demonstrate the algorithm requires well-aligned and registered images to be effective. The effects due to out-of-plane misregistration were not tested, but it is likely that algorithm sensitivity would be comparable to the in-plane case. If it is known

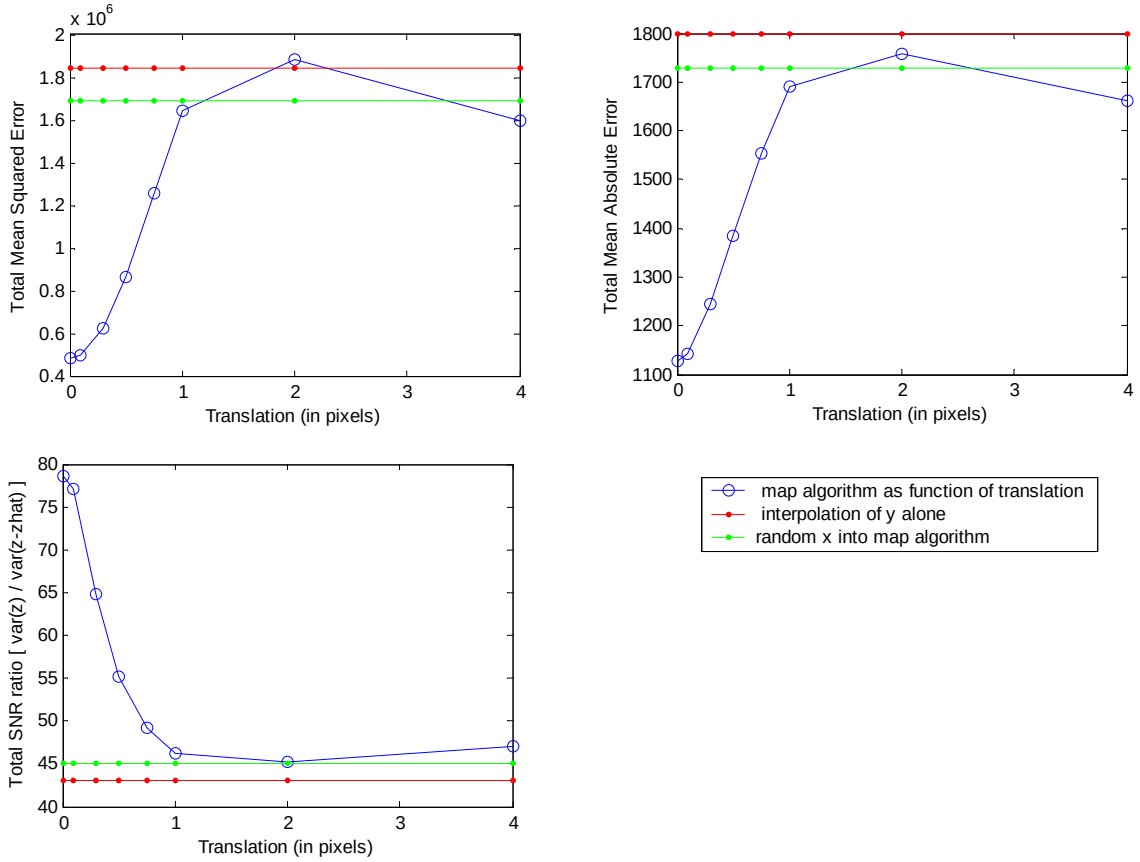


Figure 8.2: Translation misregistration

that target sensor platform can fall out of alignment, it is necessary to add some front-end preprocessing to register the images.

Bibliography

[1] R. Nishii, S. Kusanobu, and S. Tanaka, "Enhancement of low resolution image based on high resolution bands," *IEEE Tranactions on Geoscience and Remote Sensing*, vol. 34, 1151-1158 1996.

[2] J. C. Price, "Combining panchromatic and multispectral imagery from dual resolution satellite instruments," *Remote Sens. Environ.*, vol. 21, pp. 119–128, 1987.

[3] R. A. Schowengerdt, "Reconstruction of multispatial, multispectral image data using spatial frequency content," *Photogrammetric Engineering & Remote Sensing*, vol. 46, no. 10, pp. 1325–1334, 1980.

[4] W. J. Carper, T. M. Lillesand, and R. W. Kiefer, "The use of intensity-hue-saturation transformations for merging SPOT panchromatic and multispectral image data," *Photogrammetric Engineering & Remote Sensing*, vol. 56, 1990.

[5] P. S. Chavez, S. C. Sides, and J. A. Anderson, "Comparison of three different methods to merge multiresolution and multispectral data: Landsat TM and SPOT panchromatic," *Photogrammetric Engineering & Remote Sensing*, vol. 57, no. 3, pp. 295–303, 1991.

[6] V. K. Shettigara, "A generalized component substitution technique for spatial enhancement of multispectral images using a higher resolution data set," *Photogrammetric Engineering & Remote Sensing*, vol. 58, pp. 561–567, 1992.

[7] C. K. Munechika, J. S. Warnick, C. Salvaggio, and J. R. Schott, "Resolution enhancement of multispectral image data to improve classification accuracy," *Photogrammetric Engineering & Remote Sensing*, vol. 59, 67-72 1993.

[8] E. Iverson and J. R. Lersch, "Adpative image sharpening using multiresolution representations," *Proceedings of SPIE*, vol. 2231, pp. 72–83, 1994.

[9] D. P. Filberti, S. E. Marsh, and R. A. Schowengardt, "Synthesis of imagery from high spatial and spectral resolution from multiple image sources," *Optical Engineering*, vol. 33, 49-56 1994.

[10] B. Zhukov, M. Berger, F. Lanzl, and H. Kaufman, "A new technique for merging multispectral and panchromatic images revealing sub-pixel spectral variation," in *Proc IEEE IGARSS*, (Firenze, Italy), pp. 2154–2156, 1995.

[11] B. Zhukov, D. Oertel, and F. Lanzl, "A multiresolution, multisensor technique for satellite remote sensing," in *Proc IEEE IGARSS*, (Firenze, Italy), pp. 51–53, 1995.

[12] H. N. Gross and J. R. Schott, "Application of spatial resolution enhancement and spectral mixture analysis to hyperspectral images," *Proceedings of SPIE*, vol. 2821, pp. 30–41, 1996.

[13] H. N. Gross and J. R. Schott, "Application of spectral mixture analysis and image fusion techniques for image sharpening," *Remote Sens. Environ.*, vol. 63, pp. 85–94, 1998.

[14] G. D. Robinson, H. N. Gross, and J. R. Schott, "Evaluation of two applications of spectral mixing models to image fusion," *Remote Sens. Environ.*, vol. 71, pp. 272–281, 2000.

[15] M. E. Winter and E. M. Winter, "Physics-based resolution enhancement of hyperspectral data," *Proceedings of SPIE*, vol. 4725, pp. 580–587, 2002.

[16] T. Ranchin and L. Wald, "Fusion of high spatial and spectral resolution images: the ARSIS concept and its implementation," *Photogrammetric Engineering & Remote Sensing*, vol. 66, pp. 49–56, 2000.

[17] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewoods Cliffs, New Jersey: Prentice Hall, 1993.

[18] R. Hardie, M. Eismann, and G. Wilson, "Map estimation for hyperspectral image resolution enhancement using an auxiliary sensor," *IEEE Transactions on Image Processing*, vol. 13, pp. 1174–1184, Sept 2004.

[19] H. Eves, *Elementary Matrix Theory*. Dover, 1966. p.107.

[20] K. Schittkowski, "Schittkowski's implementation of powell's method, program qld.c." http://plato.la.asu.edu/topics/problems/nlores.html.

[21] B. R. Hunt and T. M. Cannon, "Nonstationary assumptions for Gaussian models of images," *6*, 876-881 1976.

[22] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantization," *IEEE Trans. Commun. Theory*, vol. 28, pp. 84–95, Jan. 1980.

[23] W. M. Porter, "Evolution of the airborne visible/infrared imaging spectrometer (AVIRIS) flight and ground data processing system," *Proceedings of the SPIE*, vol. 1298, pp. 11–17, 1990.

[24] R. C. Hardie, K. J. Barnard, J. G. Bognar, E. E. Armstrong, and E. A. Watson, "High resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system," *Optical Engineering*, vol. 37, 247-260 1998.

[25] M. T. Eismann and R. C. Hardie, "Resolution enhancement of hyperspectral imagery using coincident panchromatic imagery and a stochastic mixing model," *Proceedings of the IEEE Workshop on Land Remote Sensing*, October 2003.

[26] M. T. Eismann and R. C. Hardie, "Application of the stochastic mixing model to hyperspectral resolution enhancement," *submitted to IEEE Trans. Geoscience and Remote Sensing*, October 2003.

[27] M. T. Eismann and R. C. Hardie, "Initialization and convergence of the stochastic mixing model," *Proceedings of the SPIE*, vol. 5159, 2003.

[28] A. D. Stocker and A. P. Schaum, "Application of stochastic mixing models to hyperspectral detection problems," *Proceedings of the SPIE*, vol. 3071, pp. 47–60, 1997.

[29] P. S. Kealy and S. J. Hook, "Separating temperature and emissivity in thermal infrared multispectral scanner data: Implications for recovering land surface temperatures," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 31, Nov. 1993.

[30] A. R. Gillespie, S. Rokugawa, S. Hook, T. Matsunaga, and A. Kahle, "Temperature/emissivity separation algorithm theoretical basis document, version 2.4," *PDF file on web*, Mar. 1999. http://eospso.gsfc.nasa.gov/eos_homepage/for_scientists/atbd/docs/ASTER/atbd-ast-05-08.pdf.

[31] P. Dash, F. Gottsche, F. Olesen, and H. Fischer, "Land surface temperature and emissivity estimation from passive sensor data," *International Journal of Remote Sensing*, vol. 23, pp. 2563–2594, Jul. 20 2002.

[32] V. Payan and A. Royer, "Analysis of temperature emissivity separation (tes) algorithm applicability and sensitivity," *International Journal of Remote Sensing*, vol. 25, pp. 15–37, Jan. 2004.

[33] S. Adler-Golden, M. Matthew, L. Bernstein, R. Levine, A. Berk, S. Richtsmeier, P. Acharya, G. Anderson, G. Felde, J. Gardner, M. Hoke, L. Jeong, B. Pukall, J. Mello, A. Ratkowski, and H. Burke, "Atmospheric correction for short-wave spectral imagery based on modtran4," *SPIE Proc. Imaging Spectrometry*, vol. 3753, pp. 61–69, 1999.

[34] P. Wang, K. Liu, T. Cwik, and R. Green, "Modtran on supercomputers and parallel systems," *J. Parallel Computing*, vol. 28, pp. 53–64, Jan. 2002.

Appendix A

Analysis of the Form 2 MAP Estimate

As we saw in Section 4.5, the Form 1 and Form 2 MAP estimates are identical and equal to the expected value of $\boldsymbol{z}|\boldsymbol{\psi}$ under the assumption $\boldsymbol{x} = S\boldsymbol{z} + \boldsymbol{\eta}$. This appendix may therefore be considered as redundant, however it does present alternate derivations that might be useful at the coding level. It also serves as a check on the derivations presented in Section 4.11. The three conditions of Section 4.3 are assumed and the definitions of $B_{m,j}$ given by Equations (4.49) and (4.50) are adopted.

Starting with the Form 2 MAP estimate presented in Section 4.5, this section derives an equivalent expression that requires fewer matrix inversions (from 4 to 2). Assuming the special case where the $P \times P$ blocks of $C_{\tilde{z}}$ are identical within super-pixels, we then prove that the matrix appearing as an inverse in the new expression is singular if and only if $\sigma_n = \sigma_\eta = 0$. A separate expression is developed under the noise free and special case assumptions and in this expression which is identical to that of the previous section.

From Section 4.5, the Form 2 MAP estimate is given by:

$$\hat{\boldsymbol{z}} = \left[C_z^{-1} + W^T C_n^{-1} W + S^T C_\eta^{-1} S\right]^{-1} \left[C_z^{-1}\boldsymbol{\mu}_z + W^T C_n^{-1}\boldsymbol{y} + S^T C_\eta^{-1}\boldsymbol{x}\right] \tag{A.1}$$

Let $R = \left[C_z^{-1} + W^T C_n^{-1} W\right]^{-1}$ and use the matrix inversion lemma (see Equation (4.32)) to obtain[1] :

$$R = C_z - C_z W^T (W C_z W^T + C_n)^{-1} W C_z. \tag{A.2}$$

Apply the matrix inversion lemma again to obtain:

$$\left[R^{-1} + S^T C_\eta^{-1} S\right]^{-1} = R - RS^T (SRS^T + C_\eta)^{-1} SR.$$

[1]Comparing with Equation (4.27), we see that the matrix R is really $C_{z|y}$.

Then,

$$\begin{aligned}\hat{\boldsymbol{z}} &= \left[R - RS^T(SRS^T + C_\eta)^{-1}SR\right]\left[C_z^{-1}\boldsymbol{\mu}_z + W^T C_n^{-1}\boldsymbol{y} + S^T C_\eta^{-1}\boldsymbol{x}\right] \\ &= \left[R - RS^T(SRS^T + C_\eta)^{-1}SR\right]\left(C_z^{-1}\boldsymbol{\mu}_z + W^T C_n^{-1}\boldsymbol{y}\right) \\ &+ \left[R - RS^T(SRS^T + C_\eta)^{-1}SR\right]\left(S^T C_\eta^{-1}\boldsymbol{x}\right) \\ &= \mathbf{u} + \mathbf{v} - RS^T(SRS^T + C_\eta)^{-1}S\mathbf{u}, \end{aligned} \tag{A.3}$$

where

$$\begin{aligned}\mathbf{u} &= R\left(C_z^{-1}\boldsymbol{\mu}_z + W^T C_n^{-1}\boldsymbol{y}\right), \\ \mathbf{v} &= \left[R - RS^T(SRS^T + C_\eta)^{-1}SR\right]\left(S^T C_\eta^{-1}\boldsymbol{x}\right).\end{aligned}$$

After some algebra, new expressions for $\mathbf{u}$ and $\mathbf{v}$ may be derived [2]:

$$\mathbf{u} = \boldsymbol{\mu}_z + C_z W^T(WC_zW^T + C_n)^{-1}(\boldsymbol{y} - W\boldsymbol{\mu}_z) \tag{A.4}$$

$$\mathbf{v} = RS^T(SRS^T + C_\eta)^{-1}\boldsymbol{x}. \tag{A.5}$$

Substituting (A.5) into Equation (A.3) yields:

$$\hat{\boldsymbol{z}} = \mathbf{u} + RS^T(SRS^T + C_\eta)^{-1}(\boldsymbol{x} - S\mathbf{u}). \tag{A.6}$$

Equations (A.2), (A.4), and (A.6) taken together specify the Form 2 MAP estimate. It is now straightforward to derive analogous results in terms of the simpler quantities $\widetilde{W} = WQ^T$, $\widetilde{S} = SQ^T$, $\widetilde{R} = QRQ^T$, $C_{\tilde{z}} = QC_zQ^T$, $\tilde{\mathbf{u}} = Q\mathbf{u}$, $\boldsymbol{\mu}_{\tilde{z}} = Q\boldsymbol{\mu}_z$, $\tilde{\boldsymbol{z}} = Q\hat{\boldsymbol{z}}$, and $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)^T$:

$$\widetilde{R} = C_{\tilde{z}} - C_{\tilde{z}}\widetilde{W}^T(\widetilde{W}C_{\tilde{z}}\widetilde{W}^T + C_n)^{-1}\widetilde{W}C_{\tilde{z}}, \tag{A.7}$$

$$\tilde{\mathbf{u}} = \boldsymbol{\mu}_{\tilde{z}} + C_{\tilde{z}}\widetilde{W}^T(\widetilde{W}C_{\tilde{z}}\widetilde{W}^T + C_n)^{-1}(\boldsymbol{y} - \widetilde{W}\boldsymbol{\mu}_{\tilde{z}}), \tag{A.8}$$

$$\tilde{\boldsymbol{z}} = \tilde{\mathbf{u}} + \widetilde{R}\widetilde{S}^T(\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta)^{-1}(\boldsymbol{x} - \widetilde{S}\tilde{\mathbf{u}}). \tag{A.9}$$

Combining Equations (A.7), (A.8), and (A.9) and collecting terms yields:

$$\tilde{\boldsymbol{z}} = (I_{NP} - A^{(2)}\widetilde{W})(I_{NP} - \Theta\widetilde{S})\boldsymbol{\mu}_{\tilde{z}} + (I_{NP} - \Theta\widetilde{S})A^{(2)}\boldsymbol{y} + \Theta\boldsymbol{x}, \tag{A.10}$$

where the $NP \times MP$ matrix $A^{(2)}$ and the $NP \times N$ matrix Θ are given by:

$$\begin{aligned}A^{(2)} &= C_{\tilde{z}}\widetilde{W}^T(\widetilde{W}C_{\tilde{z}}\widetilde{W}^T + C_n)^{-1}, \\ \Theta &= \widetilde{R}\widetilde{S}^T(\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta)^{-1}.\end{aligned}$$

Comparing Equations (A.1) and (A.10), we see that the number of matrix inversions has been reduced from 4 to 2. Since the two eliminated are inverses of C_n and C_η, this is advantageous when either C_n or C_η is not taken as a multiple of the identity matrix. In

[2] Comparing with Equation (4.26), we see that the vector u is really $\boldsymbol{\mu}_{z|y}$.

analogy with the Form 1 MAP estimate, the Form 2 MAP estimate first computes $\tilde{\boldsymbol{z}}$ using Equation (A.10) and then computes $\hat{\boldsymbol{z}}$ from $\hat{\boldsymbol{z}} = Q^T\tilde{\boldsymbol{z}}$.

Analogous to Equation (4.61) of Section 4.11, let

$$\bar{B}_m \;=\; \sum_{j=1}^{L} \hat{\omega}_{m,j}^2 B_{m,j}, \tag{A.11}$$

where $B_{m,n}$ is the $P \times P$ matrix given by Equations (4.49) and (4.50).

The results of Section 4.11 (with $C_{\tilde{z}|x}$ replaced by $C_{\tilde{z}}$) prove that:

$$A^{(2)} \;=\; C_{\tilde{z}}\widetilde{W}^T(\widetilde{W}C_{\tilde{z}}\widetilde{W}^T + C_n)^{-1} \;=\; \bigoplus_{m=1}^{M} A_m^{(2)}, \tag{A.12}$$

where $A_m^{(2)}$ is the $LP \times P$ matrix given by:

$$A_m^{(2)} = \begin{bmatrix} \hat{\omega}_{m,1}B_{m,1}(\bar{B}_m + \sigma_n^2 I_P)^{-1} \\ \hat{\omega}_{m,2}B_{m,2}(\bar{B}_m + \sigma_n^2 I_P)^{-1} \\ \vdots \\ \hat{\omega}_{m,L}B_{m,L}(\bar{B}_m + \sigma_n^2 I_P)^{-1} \end{bmatrix}. \tag{A.13}$$

From (4.10), (A.7), and (A.12):

$$\widetilde{R} = \left[I_{NP} - \bigoplus_{m=1}^{M} A_m^{(2)} W_m \right] C_{\tilde{z}}. \tag{A.14}$$

It is readily seen from Equation (A.14) that $\widetilde{R}$ is block diagonal with $LP \times LP$ blocks. However, the $LP \times LP$ matrix $A_m^{(2)}W_m$ is *not* block diagonal with $P \times P$ blocks which implies that $\widetilde{R}$ is *not* block diagonal with $P \times P$ blocks. Nevertheless, we will still be able to establish a super-pixel breakdown of the Form 2 MAP estimate, valid under the special case assumption which will be similar to the Form 1 breakdown given by Equation (4.67).

Notice that $\bar{B}_m$ is positive definite since each $B_{m,j}$ is positive definite. This implies that the matrices $\bar{B}_m + \sigma_n I_P$ appearing in Equation (A.13) are nonsingular regardless of whether $\sigma_n^2 = 0$ or $\sigma_n > 0$, proving that $\widetilde{W}C_{\tilde{z}}\widetilde{W}^T + C_n$ is nonsingular regardless of whether $\sigma_n = 0$ or $\sigma_n > 0$. However, the inverse of $\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ also appears in the Form 2 MAP estimate as specified by Equation (A.10). We will prove below that under the special case assumption, the singular or nonsingular status of $\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ depends on the zero or nonzero status of both σ_η and σ_n. With this in mind we prove the following preliminary results, which will also be utilized when the Form 3 MAP estimate is analyzed. As before, γ_m denotes the inner product $\vec{\omega}_m^T\vec{\omega}_m$.

Lemma A.0.1 $\|\Lambda_m\|_F < 1$ *whenever* $\sigma_\eta > 0$ *or* $\sigma_n > 0$ *where*

$$\Lambda_m = \frac{\mathbf{s}^T B_{m,1} \left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}\mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2} \left(\vec{\omega}_m \vec{\omega}_m^T\right), \tag{A.15}$$

and $\|\|_F$ *denotes the Frobenius norm.*

Proof. Since $\gamma_m = \vec{\omega}_m^T \vec{\omega}_m = \|\vec{\omega}_m \vec{\omega}_m^T\|_F$,

$$\|\Lambda_m\|_F = \frac{\mathbf{s}^T B_{m,1} \left(B_{m,1} + \delta_m I_P\right)^{-1} B_{m,1}\mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2},$$

where $\delta_m = \sigma_n^2/\gamma_m$. Adding and subtracting $\delta_m I_P$ gives the following:

$$\begin{aligned} B_{m,1} \left(B_{m,1} + \delta_m I_P\right)^{-1} &= \left[-\delta_m I_P + \left(B_{m,1} + \delta_m I_P\right)\right] \left(B_{m,1} + \delta_m I_P\right)^{-1} \\ &= -\delta_m \left(B_{m,1} + \delta_m I_P\right)^{-1} + I_P. \end{aligned}$$

Therefore,

$$\begin{aligned} \|\Lambda_m\|_F &= \frac{\mathbf{s}^T \left[-\delta_m \left(B_{m,1} + \delta_m I_P\right)^{-1} + I_P\right] B_{m,1}\mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2}, \\ &= \frac{-\delta_m \mathbf{s}^T \left(B_{m,1} + \delta_m I_P\right)^{-1} B_{m,1}\mathbf{s} + \mathbf{s}^T B_{m,1}\mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2}, \end{aligned}$$

and $\|\Lambda_m\|_F < 1$ if and only if:

$$-\delta_m \mathbf{s}^T \left(B_{m,1} + \delta_m I_P\right)^{-1} B_{m,1}\mathbf{s} < \sigma_\eta^2.$$

Since $\delta_m \geq 0$, we see that this last inequality holds if $\sigma_\eta > 0$. It also holds if $\sigma_n > 0$ since in this case $\delta_m > 0$. ∎

Lemma A.0.2 $I_L - \Lambda_m$ *is singular if and only if* $\sigma_n = \sigma_\eta = 0$.

Proof. By Lemma A.0.1, it is enough to prove that $I_L - \Lambda_m$ is singular whenever $\sigma_n = \sigma_\eta = 0$. Assuming $\sigma_n = \sigma_\eta = 0$, we have from Equation (A.15) that

$$\begin{aligned} \Lambda_m &= \frac{\mathbf{s}^T B_{m,1} \left(\gamma_m B_{m,1}\right)^{-1} B_{m,1}\mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s}} \left(\vec{\omega}_m \vec{\omega}_m^T\right), \\ &= \frac{1}{\gamma_m} \vec{\omega}_m \vec{\omega}_m^T. \end{aligned}$$

Therefore,

$$I_L - \Lambda_m = \frac{1}{\gamma_m}(\gamma_m I_L - \vec{\omega}_m \vec{\omega}_m^T).$$

Since γ_m is an eigenvalue of $\vec{\omega}_m \vec{\omega}_m^T$ (with eigenvector $\vec{\omega}_m$), it follows that the determinant of $I_L - \Lambda_m$ is zero making $I_L - \Lambda_m$ singular. ∎

The following theorem establishes sufficient conditions under which all matrices appearing as inverses in the Form 2 MAP estimate of Equation (A.10) are nonsingular.

Theorem A.0.3 *For the special case $B_{m,j} = B_{m,1}$ $(1 \le j \le L,\ 1 \le m \le M)$, the matrix $H = \widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ appearing as an inverse in the Form 2 MAP estimate is nonsingular when either $\sigma_n > 0$ or $\sigma_\eta > 0$, and is singular if $\sigma_n = \sigma_\eta = 0$.*

Proof. Since the matrix $\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ is block diagonal with $LP \times LP$ blocks, its invertibility is determined from the invertibility of each block. From Equation (A.14), the m^{th} diagonal block of $\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ is given by:

$$\begin{aligned} H_m &= \widetilde{S}_L(I_{LP} - A_m^{(2)}W_m)\widehat{B}_m\widetilde{S}_L^T + \sigma_\eta^2 I_L, \\ &= \widetilde{S}_L\widehat{B}_m\widetilde{S}_L^T + \sigma_\eta^2 I_L - \widetilde{S}_L A_m^{(2)}W_m\widehat{B}_m\widetilde{S}_L^T, \end{aligned} \tag{A.16}$$

and we wish prove that $\sigma_n = \sigma_\eta = 0$ is equivalent to H_m being singular for $m = 1, 2, \ldots, M$.

Since under the special case assumption,

$$\hat{B}_m = \bigoplus_{j=1}^{L} B_{m,j} = I_L \otimes B_{m,1},$$

it easily follows (using $\widetilde{S}_L = I_L \otimes \mathbf{s}^T$) that:

$$\begin{aligned} \hat{B}_m\widetilde{S}_L^T &= I_L \otimes B_{m,1}\,\mathbf{s} \\ \widetilde{S}_L\hat{B}_m\widetilde{S}_L^T &= \left(\mathbf{s}^T B_{m,1}\mathbf{s}\right) I_L, \\ \widetilde{S}_L\hat{B}_m\widetilde{S}_L^T + \sigma_\eta^2 I_L &= \left(\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2\right) I_L. \end{aligned}$$

Likewise, from Equations (A.11) and (A.13):

$$\begin{aligned} \bar{B}_m &= \gamma_m B_{m,1} \quad \text{with} \quad \gamma_m = \vec{\omega}_m^T\vec{\omega}_m, \\ A_m^{(2)} &= \vec{\omega}_m \otimes \left[B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1}\right], \end{aligned}$$

it easily follows (using $W_m = \vec{\omega}_m^T \otimes I_P$) that:

$$A_m^{(2)}W_m = \vec{\omega}_m\vec{\omega}_m^T \otimes B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1}.$$

Thus,

$$\begin{aligned} A_m^{(2)}W_m\hat{B}_m &= \vec{\omega}_m\vec{\omega}_m^T \otimes B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}, \\ \widetilde{S}_L A_m^{(2)}W_m\hat{B}_m\widetilde{S}_L^T &= \vec{\omega}_m\vec{\omega}_m^T \otimes \left(\mathbf{s}^T B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}\mathbf{s}\right), \\ &= \left(\mathbf{s}^T B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}\mathbf{s}\right)\vec{\omega}_m\vec{\omega}_m^T, \end{aligned}$$

and substitution into Equation (A.16) yields:

$$H_m = \left(\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_n^2\right) I_L - \left(\mathbf{s}^T B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}\mathbf{s}\right)\vec{\omega}_m\vec{\omega}_m^T.$$

Therefore,

$$H_m \;=\; \left(\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2\right)\left(I_L - \Lambda_m\right),$$

where Λ_m is given by Equation (A.15). By Lemma A.0.2, the theorem is proven. ∎

As we will see in Appendix B, the proof of Theorem B.0.5 shows that $I_L - \Lambda_m$ is positive definite. This implies that H_m and hence $H = \widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ is positive definite under the special case assumption and the assumption that either $\sigma_n > 0$ or $\sigma_\eta > 0$.

Below, the Form 2 MAP estimate given by Equation (A.10) is broken down into super-pixel components, making it amenable to parallel processing. By Theorem A.0.3, it is valid under the special case assumption and the assumption that either $\sigma_n > 0$ or $\sigma_\eta > 0$. As with super-pixel breakdown of the Form 1 MAP estimate, the $LP \times 1$ super-pixel components are denoted by $\tilde{\boldsymbol{z}}_m$ and are given in terms of the $P \times 1$ low resolution hyper-pixel $\vec{y}_m$. Also appearing is the $L \times 1$ vector $\boldsymbol{x}_m$. The lack of a tilde on $\boldsymbol{x}_m$ is intentional since it represents the m^{th} set of L consecutive entries of $\boldsymbol{x}$ itself and not a permuted version of $\boldsymbol{x}$. From Equation (A.10), the m^{th} super-pixel Form 2 MAP estimate is given by:

$$\tilde{\boldsymbol{z}}_m \;=\; \left(I_{LP} - A_m^{(2)} W_m\right)\left(I_{LP} - \Theta_m \widetilde{S}_L\right)\boldsymbol{\mu}_{\tilde{z}_m} + \left(I_{LP} - \Theta_m \widetilde{S}_L\right) A_m^{(2)} \vec{y}_m + \Theta_m \boldsymbol{x}_m \quad \text{(A.17)}$$

where

$$\Theta \;=\; \widetilde{R}\widetilde{S}^T(\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta)^{-1} \;=\; \bigoplus_{m=1}^{M} \Theta_m.$$

From equation (A.14):

$$\begin{aligned} \Theta_m &= (I_{LP} - A_m^{(2)} W_m)\widehat{B}_m \widetilde{S}_L^T H_m^{-1}, \\ H_m &= \widetilde{S}_L (I_{LP} - A_m^{(2)} W_m)\widehat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L. \end{aligned}$$

The special case assumption provides significant simplifications of H_m, Θ_m, and $A_m^{(2)}$:

$$H_m \;=\; \left(\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2\right)\left(I_L - \Lambda_m\right), \quad \text{(A.18)}$$

$$\Theta_m \;=\; \left(I_L \otimes B_{m,1}\mathbf{s} - \vec{\omega}_m \vec{\omega}_m^T \otimes C_m \mathbf{s}\right) H_m^{-1}, \quad \text{(A.19)}$$

$$A_m^{(2)} \;=\; \vec{\omega}_m \otimes \left[B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1}\right]. \quad \text{(A.20)}$$

where

$$\Lambda_m \;=\; \frac{\mathbf{s}^T C_m \mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2}\left(\vec{\omega}_m \vec{\omega}_m^T\right), \quad \text{(A.21)}$$

$$C_m \;=\; B_{m,1}\left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}. \quad \text{(A.22)}$$

Alternate expressions for $A_m^{(2)}$ and C_m are more efficient computationally:

$$\begin{aligned} A_m^{(2)} &= \vec{\omega}_m \otimes \frac{1}{\gamma_m}\left[I_P - \delta_m\left(B_{m,1} + \delta_m I_P\right)^{-1}\right], \\ C_m &= \frac{1}{\gamma_m}\left(B_{m,1} - \delta_m I_P + \delta_m^2\left(B_{m,1} + \delta_m I_P\right)^{-1}\right). \end{aligned}$$

where $\delta_m = \sigma_n^2/\gamma_m$. These may be derived from Equations (A.20) and (A.22) by adding and subtracting $\delta_m I_P$ from $B_{m,1}$ and distributing the factor $(B_{m,1} + \delta_m I_P)^{-1}$.

Equation (A.17) is valid whenever either $\sigma_n > 0$ or $\sigma_\eta > 0$ and the $P \times P$ blocks of $C_{\tilde{z}}$ are assumed to be identical within super-pixels. As was the case for the Form 1 MAP estimate, the computation of $\tilde{\boldsymbol{z}}_m$ using Equation (A.17) may be accomplished in parallel by assigning a different set of m to each processing unit, thereby avoiding the need for parallel matrix algorithms.

It was shown in Theorem A.0.3 that H_m is singular when $\sigma_\eta = \sigma_n = 0$, which would seem to indicate that the noise free case is incompatible with the Form 2 MAP estimate. However, as we prove below, the singularity represented in Θ_m cancels in the noise free case, allowing the estimate to be expressed in a way that no inverse matrix appears. The following lemma facilitates the argument used.

Lemma A.0.4 *For any β with $0 < \beta < \frac{1}{\gamma_m}$,*

$$(I_L - \frac{1}{\gamma_m}\Omega_m)(I_L - \beta\Omega_m)^{-1} \;=\; I_L - \frac{1}{\gamma_m}\Omega_m,$$

where $\Omega_m = \vec{\omega}_m\vec{\omega}_m^T$.

Proof. For $0 \leq \beta < \frac{1}{\gamma_m}$,

$$\|\beta\Omega_m\|_F = \beta\|\Omega_m\|_F = \beta\gamma_m < 1,$$

proving that $I_L - \beta\Omega_m$ is nonsingular. Therefore, it is equivalent to prove that:

$$(I_L - \frac{1}{\gamma_m}\Omega_m) \;=\; (I_L - \frac{1}{\gamma_m}\Omega_m)(I_L - \beta\Omega_m),$$

Multiplying this out and cancelling terms shows that the above condition is equivalent to $\frac{\Omega_m}{\gamma_m}$ being a projection map:

$$\left(\frac{\Omega_m}{\gamma_m}\right)^2 \;=\; \frac{\Omega_m}{\gamma_m},$$

which is easily verified. ∎

We will conclude the section by deriving a single expression for $\tilde{\boldsymbol{z}}$ that is free of any matrix inverses and applies under the special case and noise free assumptions. To accomplish this, we take $\sigma_n = 0$ in Equations (A.18) to (A.22):

$$\begin{aligned}
C_m &= \frac{1}{\gamma_m}B_{m,1}, \\
A_m^{(2)} &= \vec{\omega}_m \otimes \frac{1}{\gamma_m}I_P, \\
\Lambda_m &= \beta_m\vec{\omega}_m\vec{\omega}_m^T, \\
H_m &= d_m\left(I_L - \beta_m\vec{\omega}_m\vec{\omega}_m^T\right),
\end{aligned}$$

where

$$\begin{aligned} d_m &= \mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2, \\ \beta_m &= \frac{d_m - \sigma_\eta^2}{d_m \gamma_m}. \end{aligned}$$

By Lemma A.0.4, when $\sigma_n = 0$ and $\sigma_\eta > 0$, Equation (A.19) becomes:

$$\begin{aligned} \Theta_m &= \frac{1}{d_m}\left(I_L \otimes B_{m,1}\mathbf{s} - \vec{\omega}_m\vec{\omega}_m^T \otimes \frac{1}{\gamma_m}B_{m,1}\mathbf{s}\right)\left(I_L - \beta_m\vec{\omega}_m\vec{\omega}_m^T\right)^{-1}, \\ &= \frac{1}{d_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right]\left(I_L - \beta_m\vec{\omega}_m\vec{\omega}_m^T\right)^{-1}, \\ &= \frac{1}{d_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right)\left(I_L - \beta_m\vec{\omega}_m\vec{\omega}_m^T\right)^{-1}\right] \otimes B_{m,1}\mathbf{s} \\ &= \frac{1}{d_m}\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}. \end{aligned}$$

It follows that as $\sigma_\eta \to 0$:

$$\Theta_m \longrightarrow \frac{1}{a_m}\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s},$$

where $a_m = \mathbf{s}^T B_{m,1}\mathbf{s}$.

Substituting these results into Equation (A.17) and simplifying yields the super-pixel breakdown of the Form 2 MAP estimate valid under the noise free special assumptions:

$$\begin{aligned} \tilde{\boldsymbol{z}}_m &= \left[(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T) \otimes (I_P - \frac{1}{a_m}B_{m,1}\mathbf{s}\mathbf{s}^T)\right]\boldsymbol{\mu}_{\tilde{z}_m} \\ &+ \frac{1}{a_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right]\boldsymbol{x}_m + \frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P)\vec{y}_m. \end{aligned} \tag{A.23}$$

As a check, we see that Equations (4.70) and (A.23) are identical.

Appendix B

Analysis of the Form 3 MAP Estimate

The analysis of the Form 3 MAP estimate presented in this appendix is analogous to that of the Form 1 and Form 2 MAP estimate presented in Section 4.11 and Appendix A. We first express the estimate in terms of simpler matrix quantities by applying the permutation matrix Q. Then, we break down the coefficient matrix of the system into super-pixel blocks, and analyze the special case where the $P \times P$ blocks of the covariance matrix $C_{\tilde{z}}$ are identical within super-pixels. The section concludes by providing a simplified expression for the Form 3 MAP estimate broken down into super-pixel components, applicable under the special case nose free assumption. Significant use is made of the properties of the Kronecker tensor product listed in the Chapter 4.

Starting with Equation (4.39) of Section 4.5, we apply the permutation matrix Q to derive the equivalent expression:

$$\tilde{\boldsymbol{z}} \;=\; \boldsymbol{\mu}_{\tilde{z}} + A^{(3)} \left[\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix} - \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix} \boldsymbol{\mu}_{\tilde{z}} \right], \tag{B.1}$$

where

$$A^{(3)} \;=\; C_{\tilde{z}} \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix}^T \left[\begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix} C_{\tilde{z}} \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix}^T + \begin{pmatrix} C_n & 0 \\ 0 & C_\eta \end{pmatrix} \right]^{-1}.$$

Similar to the Form 2 MAP estimate, we examine the conditions under which:

$$H \;=\; \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix} C_{\tilde{z}} \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix}^T + \begin{pmatrix} C_n & 0 \\ 0 & C_\eta \end{pmatrix} \tag{B.2}$$

is singular or nonsingular. The three conditions of Section 4.3 are assumed and the definitions of $B_{m,j}$, and $\bar{B}_m$ given by Equations (4.49), (4.50), (A.11) are adopted. Thus,

$$\begin{aligned}\widetilde{S} &= I_M \otimes \widetilde{S}_L = \bigoplus_{m=1}^{M} \widetilde{S}_L, \\ \widetilde{S}_L &= I_L \otimes \mathbf{s}^T = \bigoplus_{j=1}^{L} \mathbf{s}^T,\end{aligned}$$

and

$$\widetilde{W} = \bigoplus_{m=1}^{M} \left(\vec{\omega}_m^T \otimes I_P\right).$$

From equation (B.2),

$$H = \begin{bmatrix} \widetilde{W} C_{\tilde{z}} \widetilde{W}^T + C_n & \widetilde{W} C_{\tilde{z}} \widetilde{S}^T \\ \widetilde{S} C_{\tilde{z}} \widetilde{W}^T & \widetilde{S} C_{\tilde{z}} \widetilde{S}^T + C_\eta \end{bmatrix}.$$

Using

$$\begin{aligned}\widetilde{W} C_{\tilde{z}} \widetilde{W}^T &= \bigoplus_{m=1}^{M} \bar{B}_m, \\ \widetilde{S} C_{\tilde{z}} \widetilde{S}^T &= \bigoplus_{m=1}^{M} \widetilde{S}_L \hat{B}_m \widetilde{S}_L^T, \\ \widetilde{W} C_{\tilde{z}} \widetilde{S}^T &= \bigoplus_{m=1}^{M} (\vec{\omega}_m^T \otimes I_P) \hat{B}_m \widetilde{S}_L^T,\end{aligned}$$

we break down H into blocks:

$$H = \begin{bmatrix} \bigoplus_{m=1}^{M} \left(\bar{B}_m + \sigma_n^2 I_P\right) & \bigoplus_{m=1}^{M} \left((\vec{\omega}_m^T \otimes I_P) \hat{B}_m \widetilde{S}_L^T\right) \\ \bigoplus_{m=1}^{M} \left(\widetilde{S}_L \hat{B}_m (\vec{\omega}_m \otimes I_P)\right) & \bigoplus_{m=1}^{M} \left(\widetilde{S}_L \hat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L\right) \end{bmatrix}.$$

Since both $\bar{B}_m + \sigma_n^2 I_P$ and $\widetilde{S}_L \hat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L$ are positive definite matrices, we may define:

$$U = U^T = \begin{bmatrix} \bigoplus_{m=1}^{M} (\bar{B}_m + \sigma_n^2 I_P)^{-1/2} & 0 \\ 0 & \bigoplus_{m=1}^{M} (\widetilde{S}_L \hat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L)^{-1/2} \end{bmatrix}, \tag{B.3}$$

so that

$$U^T H U \;=\; \begin{bmatrix} I_{MP} & \bigoplus_{m=1}^{M} F_m \\ \bigoplus_{m=1}^{M} F_m^T & I_N \end{bmatrix},$$

where

$$F_m \;=\; (\bar{B}_m + \sigma_n^2 I_P)^{-1/2} (\vec{\omega}_m^T \otimes I_P) \hat{B}_m \widetilde{S}_L^T (\widetilde{S}_L \hat{B}_m \widetilde{S}_L^T + \sigma_\eta^2 I_L)^{-1/2}. \tag{B.4}$$

Defining

$$V \;=\; \begin{bmatrix} I_{MP} & -\bigoplus_{m=1}^{M} F_m \\ 0 & I_N \end{bmatrix}, \tag{B.5}$$

we apply V^T on the left of $U^T H U$ and V on the right to obtain the block diagonal matrix D:

$$D \;=\; (UV)^T H (UV) \;=\; V^T (U^T H U) V \;=\; \begin{bmatrix} I_{MP} & 0 \\ 0 & \bigoplus_{m=1}^{M} (I_L - F_m^T F_m) \end{bmatrix}. \tag{B.6}$$

The inverse of U is obtained by replacing the exponent $-1/2$ with $1/2$ in Equation (B.3) and the inverse of V by replacing $-\oplus$ with $\oplus$ in Equation (B.5). Therefore, UV is nonsingular and we may solve for H:

$$H \;=\; \left[(UV)^T\right]^{-1} D (UV)^{-1}.$$

We see that H is nonsingular if and only if D is nonsingular and in that case:

$$H^{-1} \;=\; (UV) D^{-1} (UV)^T.$$

Thus, determining the singular or nonsingular status of H reduces to determining the singular or nonsingular status of $I_L - F_m^T F_m$ for $m = 1, 2, \ldots, M$. For the special case where the $P \times P$ diagonal blocks of $C_{\vec{z}}$ are identical within super-pixels, this allows the singular or nonsingular status of H to be characterized in terms of σ_n and σ_η as indicated in the following theorem:

Theorem B.0.5 *For the special case $B_{m,j} = B_{m,1}$ $(1 \le j \le L,\, 1 \le m \le M)$, the matrix H of Equation* (B.2) *is positive definite (hence non-singular) when either $\sigma_n > 0$ or $\sigma_\eta > 0$, and is singular whenever $\sigma_n = \sigma_\eta = 0$.*

Proof. From the development above, H is positive definite if and only if $I_L - F_m^T F_m$ is positive definite for $m = 1, 2, \ldots, M$. In general,

$$F_m^T F_m \;=\; \left(\bigoplus_{j=1}^{L} d_{m,j}^{-1/2} \mathbf{s}^T B_{m,j}\right) \left(\vec{\omega}_m \vec{\omega}_m^T \otimes (\bar{B}_m + \sigma_n^2 I_P)^{-1}\right) \left(\bigoplus_{j=1}^{L} d_{m,j}^{-1/2} B_{m,j}\mathbf{s}\right),$$

where $d_{m,j} = \mathbf{s}^T B_{m,j}\mathbf{s} + \sigma_\eta^2$. Under the special case assumption this reduces to:

$$F_m^T F_m \;=\; \frac{\mathbf{s}^T B_{m,1} \left(\gamma_m B_{m,1} + \sigma_n^2 I_P\right)^{-1} B_{m,1}\mathbf{s}}{\mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2} \left(\vec{\omega}_m \vec{\omega}_m^T\right), \tag{B.7}$$

where $\gamma_m = \vec{\omega}_m^T \vec{\omega}_m = \|\vec{\omega}_m \vec{\omega}_m^T\|_F$. Comparing with Equation (A.15), we see that $F_m^T F_m = \Lambda_m$.

To prove that $I_L - F_m^T F_m$ is positive definite, let $\mathbf{u}$ be any unit vector of length L and show

$$\mathbf{u}^T (I_L - F_m^T F_m)\mathbf{u} > 0.$$

Since, $\mathbf{u}^T(I_L - F_m^T F_m)\mathbf{u} = 1 - \|F_m\mathbf{u}\|_2^2$, this reduces to showing $\|F_m\mathbf{u}\|_2 < 1$. Using the observation that the 2-norm and Frobenius norm are identical when applied to vectors, it follows that:

$$\|F_m\mathbf{u}\|_2 \;\leq\; \|F_m\|_F \|\mathbf{u}\|_2 \;=\; \|F_m\|_F.$$

By Lemma A.0.1, $\|F_m^T F_m\|_F < 1$ and

$$\|F_m^T F_m\|_F < 1 \implies \|F_m^T\|_F \|F_m\|_F < 1 \implies \|F_m\|_F^2 < 1 \implies \|F_m\|_F < 1.$$

Therefore, $\|F_m\mathbf{u}\|_2 < 1$. ∎

In the proof of Theorem B.0.5, it is shown that $I_L - F_m^T F_m = I_L - \Lambda_m$ is positive definite. It follows that the matrix $\widetilde{S}\widetilde{R}\widetilde{S}^T + C_\eta$ appearing in the Form 2 MAP estimate (see Theorem A.0.3) not only is nonsingular under the special case assumption and the assumption that either $\sigma_n > 0$ or $\sigma_\eta > 0$, it is in fact positive definite under these assumptions.

We now derive the Form 3 MAP estimate for the noise free case ($\sigma_n = \sigma_\eta = 0$), under the special case assumption that the $P \times P$ blocks of $C_{\tilde{z}}$ are identical within super-pixels. As before, we take $\sigma_n = 0$ and examine $A^{(3)}$ in the limit as $\sigma_\eta \to 0$. Thus for the remainder of this section, we assume $\sigma_n = 0$ and $B_{m,j} = B_{m,1}$ for $1 \leq m \leq M$ and $1 \leq j \leq L$.

We successively derive expressions for D^{-1}, $VD^{-1}V^T$, $H^{-1} = U(VD^{-1}V^T)U^T$ and finally $A^{(3)} = \left[C_{\tilde{z}}\widetilde{W}^T, C_{\tilde{z}}\widetilde{S}^T\right] H^{-1}$, taking the limit as $\sigma_\eta \to 0$ in the final expression. Using

$$\begin{aligned} \bar{B}_m &= \gamma_m B_{m,1}, \\ \widetilde{S}_L &= I_L \otimes \mathbf{s}^T, \\ \hat{B}_m \widetilde{S}_L^T &= I_L \otimes B_{m,1}\mathbf{s}, \end{aligned}$$

and the properties of the Kronecker tensor product, we derive from Equation (B.4) the following expression for F_m:

$$F_m \;=\; \frac{\vec{\omega}_m^T \otimes B_{m,1}^{1/2}\mathbf{s}}{\sqrt{\gamma_m d_m}}, \tag{B.8}$$

where $d_m = \mathbf{s}^T B_{m,1}\mathbf{s} + \sigma_\eta^2$. Either of Equations (B.7) or (B.8) yields:

$$I_L - F_m^T F_m \;=\; I_L - \beta_m \left(\vec{\omega}_m \vec{\omega}_m^T\right),$$

where

$$\beta_m \;=\; \frac{d_m - \sigma_\eta^2}{\gamma_m d_m} \rightarrow \frac{1}{\gamma_m} \quad (\text{as } \sigma_\eta \rightarrow 0).$$

Defining $E_m = I_L - \beta_m \vec{\omega}_m \vec{\omega}_m^T$ we have:

$$D^{-1} \;=\; \begin{bmatrix} \bigoplus_{m=1}^{M} I_P & 0 \\ 0 & \bigoplus_{m=1}^{M} E_m^{-1} \end{bmatrix}, \text{ and}$$

$$VD^{-1}V^T \;=\; \begin{bmatrix} \bigoplus_{m=1}^{M} \left(I_P + F_m E_m^{-1} F_m^T\right) & -\bigoplus_{m=1}^{M} F_m E_m^{-1} \\ -\bigoplus_{m=1}^{M} E_m^{-1} F_m^T & \bigoplus_{m=1}^{M} E_m^{-1} \end{bmatrix}.$$

Since $U = U^T$ is given by:

$$U \;=\; \begin{bmatrix} \bigoplus_{m=1}^{M} \gamma_m^{-1/2} B_{m,1}^{-1/2} & 0 \\ 0 & \bigoplus_{m=1}^{M} d_m^{-1/2} I_L \end{bmatrix},$$

it follows that $H^{-1} = U(VD^{-1}V^T)U^T$ is given by:

$$H^{-1} \;=\; \begin{bmatrix} \bigoplus_{m=1}^{M} \frac{1}{\gamma_m} B_{m,1}^{-1/2} \left(I_P + F_m E_m^{-1} F_m^T\right) B_{m,1}^{-1/2} & -\bigoplus_{m=1}^{M} \frac{1}{\sqrt{\gamma_m d_m}} B_{m,1}^{-1/2} F_m E_m^{-1} \\ -\bigoplus_{m=1}^{M} \frac{1}{\sqrt{\gamma_m d_m}} E_m^{-1} F_m^T B_{m,1}^{-1/2} & \bigoplus_{m=1}^{M} \frac{1}{d_m} E_m^{-1} \end{bmatrix}.$$

Using Equation (B.8) and properties of the Kronecker tensor product, this reduces to:

$$H^{-1} = \begin{bmatrix} \bigoplus_{m=1}^{M} \frac{1}{\gamma_m}\left[B_{m,1}^{-1} + \frac{1}{\gamma_m d_m}\left(\vec{\omega}_m^T \otimes \mathbf{s}\right) E_m^{-1}\left(\vec{\omega}_m \otimes \mathbf{s}^T\right)\right] & -\bigoplus_{m=1}^{M} \frac{1}{\gamma_m d_m}\left(\vec{\omega}_m^T \otimes \mathbf{s}\right) E_m^{-1} \\ -\bigoplus_{m=1}^{M} \frac{1}{\gamma_m d_m} E_m^{-1}\left(\vec{\omega}_m \otimes \mathbf{s}^T\right) & \bigoplus_{m=1}^{M} \frac{1}{d_m} E_m^{-1} \end{bmatrix}.$$

In order facilitate the matrix algebra, we partition $A^{(3)}$ into blocks $A_1^{(3)}$ of size $NP \times MP$ and $A_2^{(3)}$ of size $NP \times N$:

$$A^{(3)} = C_{\tilde{z}} \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix}^T H^{-1} = \left[A_1^{(3)}, A_2^{(3)}\right].$$

Since

$$C_{\tilde{z}} \begin{pmatrix} \widetilde{W} \\ \widetilde{S} \end{pmatrix}^T = \left[\bigoplus_{m=1}^{M} \left(\vec{\omega}_m \otimes B_{m,1}\right), \bigoplus_{m=1}^{M} \left(I_L \otimes B_{m,1}\mathbf{s}\right)\right],$$

we have:

$$\begin{aligned} A_1^{(3)} &= \bigoplus_{m=1}^{M} \left\{ \frac{1}{\gamma_m}\left(\vec{\omega}_m \otimes B_{m,1}\right)\left[B_{m,1}^{-1} + \frac{1}{\gamma_m d_m}\left(\vec{\omega}_m^T \otimes \mathbf{s}\right) E_m^{-1}\left(\vec{\omega}_m \otimes \mathbf{s}^T\right)\right] \right. \\ &- \left. \frac{1}{\gamma_m d_m}\left(I_L \otimes B_{m,1}\mathbf{s}\right) E_m^{-1}\left(\vec{\omega}_m \otimes \mathbf{s}^T\right) \right\}, \\ A_2^{(3)} &= \bigoplus_{m=1}^{M} \left\{ -\frac{1}{\gamma_m d_m}\left(\vec{\omega}_m \otimes B_{m,1}\right)\left(\vec{\omega}_m^T \otimes \mathbf{s}\right) E_m^{-1} + \frac{1}{d_m}\left(I_L \otimes B_{m,1}\mathbf{s}\right) E_m^{-1} \right\}. \end{aligned}$$

Applying the properties of the Kronecker tensor product, these reduce to the following:

$$\begin{aligned} A_1^{(3)} &= \bigoplus_{m=1}^{M} \frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P) - \frac{1}{\gamma_m d_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) E_m^{-1} \otimes B_{m,1}\mathbf{s}\right]\left(\vec{\omega}_m \otimes \mathbf{s}^T\right) \\ A_2^{(3)} &= \bigoplus_{m=1}^{M} \frac{1}{d_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) E_m^{-1} \otimes B_{m,1}\mathbf{s}\right]. \end{aligned}$$

By Lemma A.0.4:

$$\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) E_m^{-1} = I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T \quad (\text{for } \sigma_\eta > 0).$$

Therefore, for $\sigma_\eta > 0$, we have:

$$\begin{aligned} A_1^{(3)} &= \bigoplus_{m=1}^{M} \frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P) - \frac{1}{\gamma_m d_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right]\left(\vec{\omega}_m \otimes \mathbf{s}^T\right) \\ &= \bigoplus_{m=1}^{M} \frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P) \\ A_2^{(3)} &= \bigoplus_{m=1}^{M} \frac{1}{d_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right]. \end{aligned}$$

Taking the limit as $\sigma_\eta \to 0$, the coefficient matrix $A^{(3)}$ under the special case and noise free assumptions is:

$$A^{(3)} = \bigoplus_{m=1}^{M}\left[\frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P), \frac{1}{a_m}\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right], \tag{B.9}$$

where $a_m = \mathbf{s}^T B_{m,1}\mathbf{s}$. Breaking Equation (B.1) into super-pixel components yields:

$$\tilde{\boldsymbol{z}}_m = \boldsymbol{\mu}_{\tilde{z}_m} + A_m^{(3)}\left[\begin{pmatrix} \vec{y}_m \\ \boldsymbol{x}_m \end{pmatrix} - \begin{pmatrix} \vec{\omega}_m^T \otimes I_P \\ I_L \otimes \mathbf{s}^T \end{pmatrix}\boldsymbol{\mu}_{\tilde{z}_m}\right],$$

where $A_m^{(3)}$ is the m^{th} component of Equation (B.9). Simplification of this yields the m^{th} super-pixel of the Form 3 MAP estimate under the special case and noise free assumptions:

$$\begin{aligned} \tilde{\boldsymbol{z}}_m &= \left[(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T) \otimes (I_P - \frac{1}{a_m}B_{m,1}\mathbf{s}\mathbf{s}^T)\right]\boldsymbol{\mu}_{\tilde{z}_m} \\ &+ \frac{1}{a_m}\left[\left(I_L - \frac{1}{\gamma_m}\vec{\omega}_m\vec{\omega}_m^T\right) \otimes B_{m,1}\mathbf{s}\right]\boldsymbol{x}_m + \frac{1}{\gamma_m}(\vec{\omega}_m \otimes I_P)\vec{y}_m. \end{aligned} \tag{B.10}$$

As a check on our work, we see that Equations (4.70), (A.23), and (B.10) are identical.

Appendix C

Temperature/Emissivity Separation and Atmospheric Correction

C.1 Introduction

This appendix contains a literature review and preliminary experiments done to compensate for atmospheric effects in hyperspectral data. This preliminary work was carried out by Precila C. Ip at Alliant Techsystems - Mission Research in Nashua, New Hampshire, 03062-1323. It is not intended to be a complete study of atmospheric effects, but rather a modest supplement to the primary effort described in the majority of the report.

C.2 Temperature/Emissivity Separation Algorithms

C.2.1 Problem Statement

The general problem of Temperature/Emissivity Separation (TES) is a challenging one. The challenge arises because it is inherently an underdetermined problem: in a multispectral or hyperspectral radiance measurement with N channels, there are N+1 unknowns. There is one unknown emissivity per channel and one surface temperature unknown. For specific cases, such as hyperspectral data collected over oceans where the emissivity of water is known, the problem is easily solved. However, for data taken over land surfaces where the emissivity of the background materials are unknown or highly variable, the problem is solvable only when additional assumptions or approximations are made concerning background materials and their emissivity. Many iterative solutions have been proposed that constrain the extra degree of freedom.

C.2.2 Literature Review

In 1993, Kealy and Hook [29] evaluated three methods for recovery of temperature and emissivity from multispectral thermal infrared (TIR) radiance spectra acquired over land surfaces. The instruments for data collection for this study were the Thermal Infrared Multispectral Scanner (TIMS) and the Advanced Spaceborne Thermal Emission Reflectance Radiometer (ASTER). ASTER is a scanning instrument on the NASA's Terra Satellite for recording thermal IR data to obtain surface temperatures and emissivity spectra. For ASTER, there are 5 channels, therefore the radiance estimation problem contains 5 measurements, but 6 unknowns when surface temperature is included. The three methods evaluated by Kealy and Hook were: (1) reference channel, (2) normalized emissivity (NEM), and (3) alpha-derived emissivity (ADE). The result of their study showed that NEM and ADE are more accurate than the reference channel method. Between NEM and ADE, ADE is superior for terrain backgrounds with widely varying material emissivity such as vegetation and igneous rocks. The ADE method is also equally accurate for data that was convolved with the filter response functions of the TIMS and ASTER instruments.

In 1994, a working group (ASTER Temperature/Emissivity Separation Algorithm Feasibility Study) was formed to evaluate ten existing algorithms for obtaining temperatures and emissivity. For each of the ten algorithms, [30] listed the reasons for rejection of the particular algorithm by the ASTER working group. The reviewed algorithms were:

1. Alpha-derived emissivity (ADE) method
2. Classification method
3. Day-night measurement
4. Emissivity bounds method
5. Graybody emissivity method
6. Mean-MMD method (MMD)
7. Model emissivity method
8. Normalized emissivity method (NEM)
9. Ratio algorithm (RAT)
10. Split-window algorithm

The members of this group consisted of experts in the field from U.S. and Japan. Since this initial meeting, the group has met regularly to develop, improve, and validate algorithms for high accuracy land surface temperature and emissivity computation. There other primary function is to distribute the products to the user community. The websites for the ASTER data products and the working group are:

http://asterweb.jpl.nasa.gov/products/data_products.htm

http://www.science.aster.ersdac.or.jp/en/science_info/te_WG.html

Available data products relevant to the work presented here are AST08 (surface kinetic temperature), AST05 (surface emissivity), and AST09T (atmospheric corrected surface radiance). These three products have been validated for cloud-free pixels.

C.2.3 The Hybrid TES Algorithm

Based on the reviews, the ASTER working group developed a hybrid algorithm based on two algorithms: NEM and MMD (the latter is based on ADE). The advantages of this hybrid TES algorithm over NEM or MMD alone are greater accuracy and precision for temperature and emissivity recovery. The goals of the hybrid TES algorithm are to obtain surface temperatures especially over vegetation, water, and snow as well as to obtain surface emissivity for mineral substrates.

In 2002, Dash et al. [31] did a comprehensive review of methods on estimating land surface temperature (LST) and emissivity using passive sensor data. He concluded that the hybrid TES algorithm is applicable and viable if information on the atmosphere and the multiple IR channels are available. The main disadvantage for using hybrid TES algorithm is that accurate determination of the downwelling atmospheric irradiance is critical to the results.

Most recently, in 2004, Payan et al. [32] evaluated the hybrid TES algorithm for both hyperspectral and multispectral data. They confirmed the applicability of the algorithm in both long-wave infrared (LWIR) and mid-wave infrared (MWIR) for shaded surfaces. However, this study [32] found that the methods cannot retrieve emissivity for high contrast surfaces such as metal. It also showed that the algorithms cannot be used to estimate emissivity for hyperspectral data in the MWIR for cases where the sun is directly illuminating the surface.

Based on the studies and review in [30–32], we concluded that the hybrid TES algorithm is the most robust for emissivity and temperature calculation in hyperspectral data. In future work, we would like to implement and exhaustively test this hybrid algorithm for various temperature/emissivity separation tasks.

The steps for implementing the TES algorithm, which is a combination of the NEM, RAT, and MMD modules are [30]:

1. Estimate the surface temperature and subtract the reflected sky irradiance iteratively (NEM). This method first removes the atmospheric radiance and then estimates the temperature and emissivity using an assumed maximum emissivity.
2. Obtain relative emissivities and spectral shape by performing a ratio of the emissivities from NEM to the average emissivity (RAT).
3. Estimate the TES emissivities and temperature using regression analysis (MMD). This method obtains the minimum emissivity empirically and then calculates the absolute emissivity.

Detailed flow charts and description of the hybrid TES algorithm is available in [30]. This algorithm may become available as a consumer off-the-shelf product in the near future.

C.2.4 Preliminary Temperature/Emissivity Separation Results

As a preliminary and exploratory experiment, we decided to verify the performance of the Reference Channel Method and the Normalized Emissivity Method used by Kealy and Hook [29] on a representative hyperspectral data cube. To perform this task, we used ENVI (the Environment for Visualizing Images) Version 3.6 to generate surface temperatures from radiance data and apply the algorithms described. It should be noted that ENVI 3.6 does not generate temperature using the Alpha Residual algorithm (the third method discussed in [29]), although it is likely that it will be incorporated into a future release of ENVI.

The hyperspectral data set utilized was taken by the NASA MODIS/ASTER Airborne Simulator (MASTER) scanner over the Cuprite Mining District in Nevada. This scanner covers 50 bands in the spectral range 0.46-12.845 microns. For retrieving surface temperature, only the 9 thermal bands from 8.2-12.845 microns are used. The results of these algorithms are shown in Figure C.1-C.2. Surface temperature (K) statistics derived using these algorithms are shown in Table C.1.

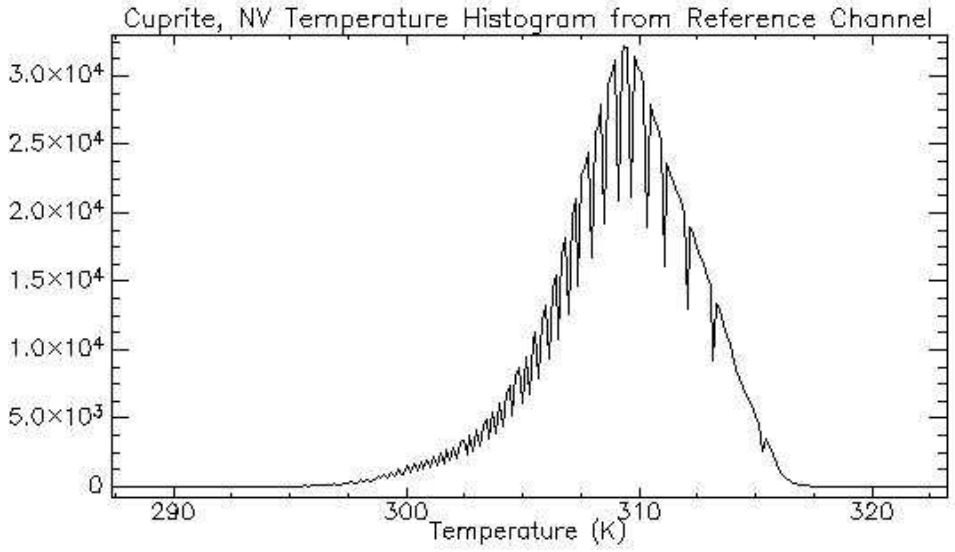


Figure C.1: The temperature histogram for a 716 by 2028 image using the Reference Channel algorithm assuming an emissivity of 0.96.

Table C.1: Derived Surface Temperature (K) Statistics versus method Algorithm

Algorithm	Min	Max	Mean	Stdev
Reference Channel	287.323212	323.351776	309.294547	3.189591
Normalized Emissivity	292.194824	331.534668	319.103284	3.652123

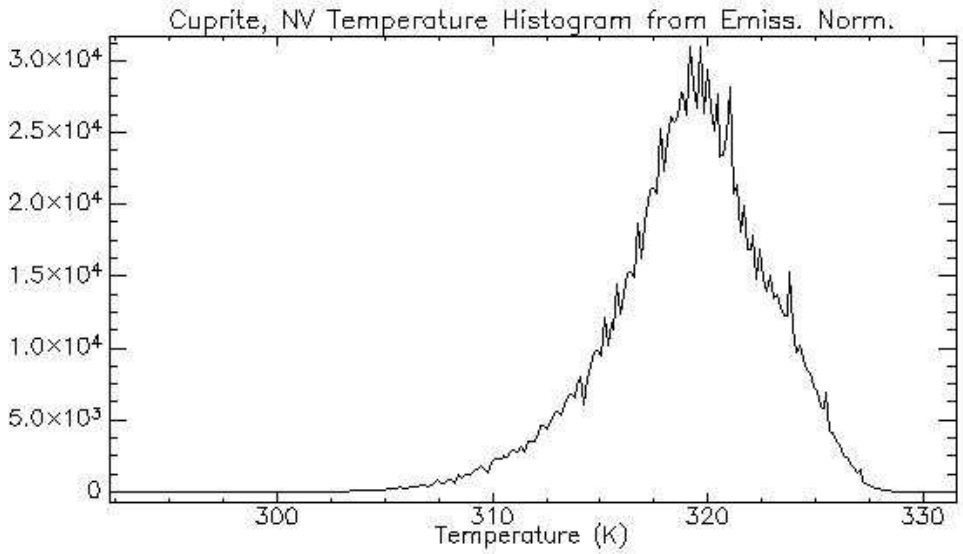


Figure C.2: The temperature histogram for a 716x2028 image using the Normalized Emissivity algorithm assuming an emissivity of 0.96.

C.3 Atmospheric Correction

In addition to investigating Temperature/Emissivity Separation Algorithms, we explored atmospheric correction algorithms that might be integrated into the MAP algorithm presented in the main body of the report. Two candidate algorithms were identified: FLAASH (Fast Line-of-sight Atmospheric Analysis of Spectral Hypercubes), and Parallel MODTRAN. The details of the algorithms are not presented here, but are available in [33] and [34].

The FLAASH algorithm may be used to retrieve reflectance data from radiance data. To demonstrate this, an representative example of AVIRIS data collected over Fort AP Hill in Virginia on September 26, 2001, was processed by FLAASH . Figure C.3 shows the radiance spectrum from 400 to 2500 nm for a vegetation pixel, and Figure C.4 shows the reflectance spectrum after FLAASH was applied for the same vegetation pixel.

For future work, there are parallel versions of FLAASH and MODTRAN 3 that might be incorporated into a parallel MAP algorithm. As a preliminary step, we implemented the NASA/JPL parallel version of MODTRAN 3 on a Beowulf Cluster. Figure C.5 shows the processing time versus the number of processors.

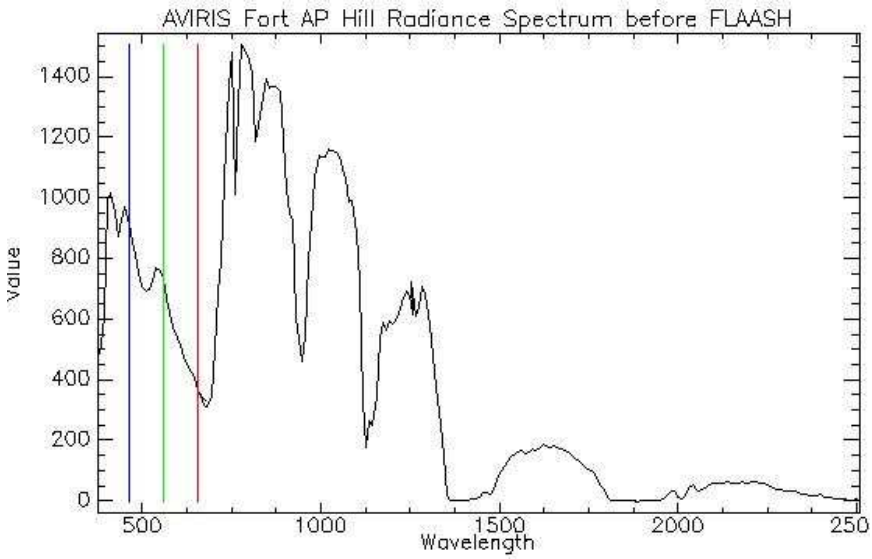


Figure C.3: AVIRIS Radiance Spectrum at Fort AP Hill.

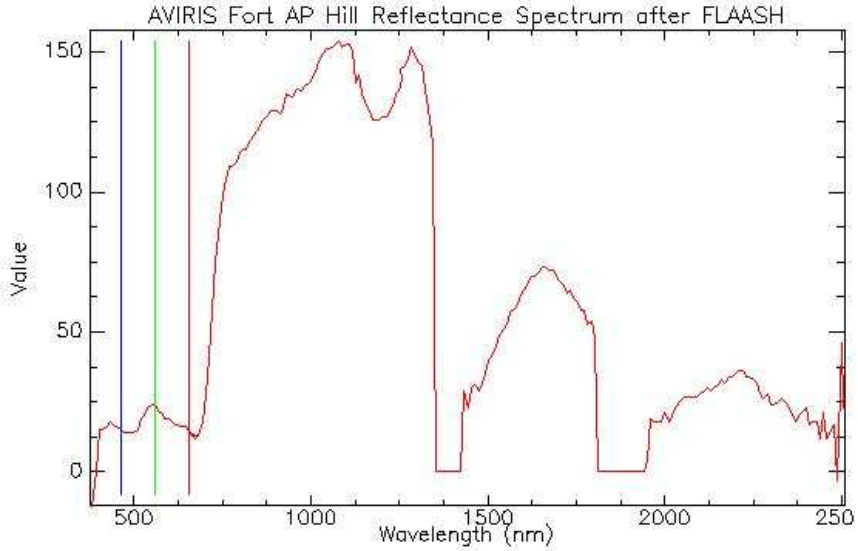


Figure C.4: AVIRIS Reflectance Spectrum at Fort AP Hill obtained from FLAASH.

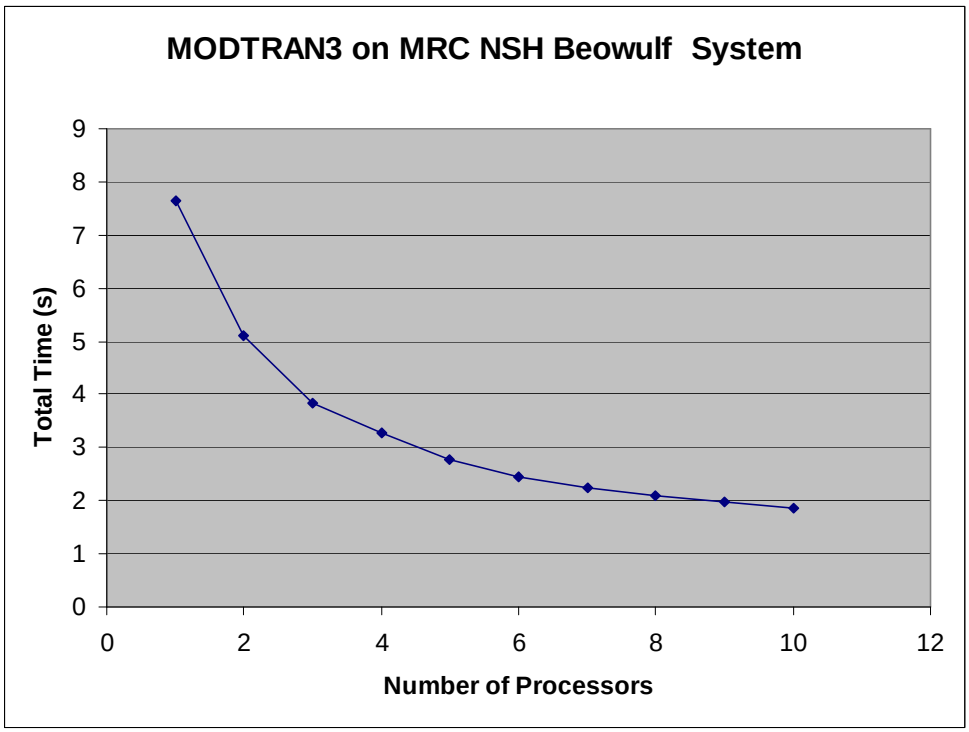


Figure C.5: Time as a function of number of processors

C.4 Summary

In this appendix, we reviewed and identified several algorithms for temperature/emissivity separation. The hybrid algorithm described in Section C.2.3 emerged as the current state-of-the-art algorithm for this purpose. For atmospheric correction, we identified parallel versions of the FLAASH and MODTRAN algorithms. Future work would include incorporating parallel implementations of the hybrid TES algorithm and FLAASH, as parts of a complete MAP enhancement of hyperspectral imagery.